xeona

An Energy Systems Modeling Environment

Core codebase
14-Oct-2010
xeona revision 5314
197 files

Suggested citation:

Morrison, Robbie. 2010. Core codebase of the *xeona* energy systems modeling environment — revision 5314. PDF version. Available as `xeona.r5314.code0.pdf`.

Robbie Morrison
Institute for Energy Engineering
Technical University of Berlin
Marchstrasse 18, D–10587 Berlin, Germany
Email: robbie@actrix.co.nz

C++ header files

- C++ language

```
./common.h
a/appinfo.h
a/exapp.h
a/exbase.h
a/exent.h
a/exitstat.h
a/floatstat.h
a/helpers.h
a/logger.h
a/recorder.h
a/xedocs.h
a/xemopt.h
a/yeek.h
b/actor.h
b/asop.h
b/asop01.h
b/asop02.h
b/asop03.h
b/auxs01.h
b/bandtaf.h
b/block.h
b/builtins.h
b/commods.h
b/commods01.h
b/costreg.h
b/domcon.h
b/entity.h
b/gate.h
b/gate01.h
b/junc.h
b/junc01.h
b/junc02.h
b/lmpbid.h
b/node.h
b/node01.h
b/node02.h
b/optctl.h
b/optgate.h
b/optjunc.h
b/optnode.h
b/optops.h
b/optprob.h
b/overseer.h
b/propdata.h
b/register.h
b/teas.h
b/teas01.h
b/teas02.h
```

```
b/teas03.h
b/teas04.h
b/teasdev.h
b/tests.h
b/tictoc.h
c/conex.h
c/costs.h
c/datio.h
c/extunits.h
c/factory.h
c/files.h
c/fincalc.h
c/ghouse.h
c/inbuilt.h
c/label.h
c/linklog.h
c/recset.h
c/si3units.h
c/simcall.h
c/stats.h
c/tsops.h
c/util1.h
c/util2.h
c/util3.h
c/xemgen.h
c/xeona_ptr.h
d/glpkviz.h
d/siglp.h
e/context.h
e/cxamb01.h
e/cxamb02.h
e/cxecon01.h
e/cxpol01.h
f/cta.h
f/gatesreg.h
f/ospinfo.h
f/trav.h
a/logger_fwd.h
a/license.h
c/smart_ptr.h
f/ospmodes.h
```

C++ implementation files

This list comprises only the core implementation files and omits the layered unit test files.

- C++ language

```
./common.cc
a/appinfo.cc
```

```
a/exapp.cc
a/exbase.cc
a/exent.cc
a/exitstat.cc
a/floatstat.cc
a/helpers.cc
a/logger.cc
a/recorder.cc
a/xedocs.cc
a/xemopt.cc
a/yeek.cc
b/actor.cc
b/asop.cc
b/asop01.cc
b/asop02.cc
b/asop03.cc
b/auxs01.cc
b/bandtaf.cc
b/block.cc
b/builtins.cc
b/commods.cc
b/commods01.cc
b/costreg.cc
b/domcon.cc
b/entity.cc
b/gate.cc
b/gate01.cc
b/junc.cc
b/junc01.cc
b/junc02.cc
b/lmpbid.cc
b/node.cc
b/node01.cc
b/node02.cc
b/optctl.cc
b/optgate.cc
b/optjunc.cc
b/optnode.cc
b/optops.cc
b/optprob.cc
b/overseer.cc

b/propdata.cc
b/register.cc
b/teas.cc
b/teas01.cc
b/teas02.cc
b/teas03.cc
b/teas04.cc
b/teasdev.cc
b/tests.cc
b/tictoc.cc
```

```
c/conex.cc
c/costs.cc
c/datio.cc
c/extunits.cc
c/factory.cc
c/files.cc
c/fincalc.cc
c/ghouse.cc
c/inbuilt.cc
c/label.cc
c/linklog.cc
c/recset.cc
c/si3units.cc
c/simcall.cc
c/stats.cc
c/tsops.cc
c/util1.cc
c/util2.cc
c/util3.cc
c/xemgen.cc
c/xeona_ptr.cc
d/glpkviz.cc
d/siglp.cc
e/context.cc
e/cxamb01.cc
e/cxamb02.cc
e/cxecon01.cc
e/cxpol01.cc
f/cta.cc
f/gatesreg.cc
f/ospinfo.cc
f/trav.cc
./main.cc
```

Ancillary files

Ancillary files provide support for building *xeona* and for variously editing, visualizing, and running *xeona* models.

- GNU make language
- GNU bash language
- GNU R language
- GNU emacs lisp language

```
./makefile
../scripts/xedoc
../scripts/mach
../scripts/xmok
../xeonar/breadplot.R
../xeonar/dataproc.R
../xeonar/extract.R
```

../xeonar/graphviz.R
../xeonar/run.R
../xeonar/saveplots.R
../xeonar/test.R
../xeonar/trial.R
../xeonar/utils.R
../xeonar/xem.R
../elisp/xeona.el
../elisp/xem.el
../elisp/xem-1.el
../elisp/xem-2.el
../elisp/xem-3.el
../elisp/xrstat.el
../elisp/xog.el
../elisp/xumber.el

◇

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : common.h
//  file-create-date : Thu 05-Apr-2007 11:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : useful definitions for the entire codebase / header
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/common.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This file declares common items that may be needed throughout
//  the application.  These various items are duly defined in
//  common.cc.
//
//  The physical design adopted removes the need for complete
//  recompilation when a variable definition is changed --
//  instead only common.o needs to be rebuilt and relinked.
//
//  This file should be hash-included in each implementation file
//  unless there is a compelling reason not to do so.  Remember
//  also to namespace-qualify these variables, for example:
//  xeona::exit_success.
//
//  An attempt to place the pre-processor macros
//  'BOOST_FILESYSTEM_NO_DEPRECATED' and
//  'BOOST_FILESYSTEM_VERSION=3' failed -- but this was only ever
//  to be a stop gap measure until version 3 became the default.

//  HEADER GUARD

#ifndef _COMMON_H_
#define _COMMON_H_

//  LOCAL AND SYSTEM INCLUDES

#include <string>             // C++ strings
#include <unistd.h>           // POSIX sleep(), usleep(), access(), chown()

// ---------------------------------------------------------
//  NAMESPACE       : xeona
// ---------------------------------------------------------

namespace xeona
{
  //  RUN-TIME FLAGS

  extern       bool again;              // set via '--again' to continue after bad solve
  extern       bool nopro;              // set via '--krazy' to exclude defensive coding
  extern       bool pepper;             // set via '--pepper' for random reseeding
  extern       bool zero;               // set via '--zero' for disabling close-to-zero
  extern       unsigned yeek;           // set via '--yeek' for selecting test code
```

```
//  DEFINITIONS: compile-time debug flag

extern const bool DBUG;               // set via xeona _XDEBUG macro
extern const bool NBUG;               // set via official NDEBUG macro

//  DEFINITIONS: current process id

extern pid_t      pid;                // underlying 'unsigned', refer <unistd.h>

//  DEFINITIONS: application exit statuses

extern const int exit_success;        // for unit testers only
extern const int exit_fail;           // for unit testers only

extern const int ret_not_overwritten;

extern const int ret_success;
extern const int ret_message;
extern const int ret_noclass;
extern const int ret_usage;
extern const int ret_fail;

extern const int exit_kill_on_log;
extern const int exit_empty_wrap;
extern const int exit_non_registration;
extern const int exit_short_timeseries;
extern const int exit_file_not_found;
extern const int exit_xem_data_issue;
extern const int exit_lazy_link_fail;
extern const int exit_cannot_run_guard;
extern const int exit_bad_authorship;
extern const int exit_full_link_fail;
extern const int exit_empty_field_on_write;
extern const int exit_timeseries_not_found;
extern const int exit_yeek_abandon;

extern const int exit_entity_fail;
extern const int exit_std_out_of_range;
extern const int exit_std_logic_error;
extern const int exit_std_bad_alloc;
extern const int exit_std_exception;
extern const int exit_bad_assign_cast;
extern const int exit_boost_exception;
extern const int exit_unknown_exception;

extern const int ret_model_file_fault;
extern const int ret_infeasibility;
extern const int ret_errant_simulation;
extern const int ret_test_code_used;
extern const int ret_other;
extern const int ret_status_not_known;

extern const int exit_test99;         // special  99 return for test code
extern const int exit_test100;        // special 100 return for test code

extern const int exitTripLevelDefault;

//  DEFINITIONS: build information

extern const std::string buildCopyright;
extern const std::string buildTitle;
extern const std::string codebaseStatus;

extern const int svnRev;              // build script macro _SVNVER or defaults to 0

extern const int gccVerMajor;         // derived from compiler macro __GNUC__
extern const int gccVerMinor;         // derived from compiler macro __GNUC_MINOR__
extern const int gccPatchLevel;       // derived from compiler macro __GNUC_PATCHLEVEL__

extern const int glpkVerMajor;        // dervived from <glpk.h> macros
extern const int glpkVerMinor;

extern const int boostVersionNum;     // derived from <boost/version.hpp> macros
extern const std::string boostVersionStr;

extern const std::string osName;      // derived from compiler macros like __linux__

extern const bool releaseStatus;      // true if _XRELEASE is defined

extern const std::string buildDate;   // derived from compiler macro __DATE__
extern const std::string buildTime;   // derived from compiler macro __TIME__
```

```
  //  DEFINITIONS: system-specific literals

  extern const char osSlash;            // pathname separator, a '/' on Linux

  //  DEFINITIONS: miscellaneous constants

  extern const std::string iso4217;          // ISO 4217 three character currency code
  extern const std::string logfileDefault;   // hardcoded logfile name instead of console

  //  DEFINITIONS: mathematical and physical constants

  extern const double mathsPi;               // maths constant 'pi'
  extern const double mathsE;                // maths constant 'e'

  extern const double stdAtmosphere;         // physics constant 'atm' (~101kPa)
  extern const double stdGravity;            // physics constant 'g'   (~9.81m/s2)
  extern const double absoluteZero;          // absolute zero          (-273.15 degrees C)

  extern const int secondsPerYear;           // seconds in 365 day year

  //  DEFINITIONS: model files

  extern const std::string backupTag;             // append tag for file backup
  extern const std::string modelExt;              // xeona model extension
  extern const std::string modelStubDefault;      // default model name
  extern const std::string modelInbuiltDefault;   // default inbuilt model name
  extern const std::string modelGuardTag;         // tag indicating guard file

  extern const std::string modelDisableChar;      // record or field disable string
  extern const std::string modelTsRepeater;       // timeseries repeat indicator
  extern const std::string modelStringDelim;      // pairwise delimiter for string values
  extern const std::string modelBidDelim;         // nodal bid separator
  extern const std::string modelEndMarker;        // model end marker

  extern const unsigned    modelFieldIndent;      // field line indent
  extern const unsigned    modelAngleIndent;      // minimum field angle char allowance

  //  DEFINITIONS: optimization problem (OSP) labeling

  extern const std::string ospTagSep;        // separator string used by class 'Label'

  //  DEFINITIONS: mandatory entity identifiers

  extern const std::string timehorizon;      // 'TimeHorizon' entity
  extern const std::string overseer;         // 'Overseer' entity

  //  DEFINITIONS: run-time behavior

  extern const unsigned reportLevelOpening;  // opening reporting level
  extern const unsigned reportLevelDefault;  // default reporting level
  extern const unsigned beepModeDefault;     // default beep behavior
  extern const unsigned sameLogLimit;        // limit repetitive log messages
  extern const unsigned consoleWidth;        // macro set via CPPFLAGS

  //  DEFINITIONS: external programs

  extern const std::string webbrowser;       // invocation string with options

  //  DEFINITIONS: xedocs text file

  extern const std::string xedocsFileName;        // xedocs file name
  extern const std::string xedocsFileContents;    // xedocs file contents

} // namespace xeona

#endif // _COMMON_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : appinfo.h
//  file-create-date : Mon 16-Apr-2007 12:07 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : start and end-of-run reporting / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/appinfo.h $

//  HEADER GUARD

#ifndef _APPINFO_H_
#define _APPINFO_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/simcall.h"     // main simulation loop

#include <string>

#include <boost/filesystem.hpp>         // path objects, iterators, useful operations
#include <boost/date_time/posix_time/ptime.hpp>   // limited header

//  CODE

// ----------------------------------------------------------
//  CLASS           : AppInfo
// ----------------------------------------------------------

class AppInfo
{
  // DISABLED

private:

  AppInfo();                                    // prevent usage

  // CREATORS

public:

  AppInfo
  (int                           argc,       // from main()
   char**                        argv,       // from main()
   const boost::filesystem::path initial);   // from main()

  ~AppInfo();

  // ACCESSORS

  const std::string
  getProgramName() const;
```

```
const std::string
getBuildDate() const;

const std::string
getBuildTime() const;

const std::string
getCodebaseStatus() const;

const std::string
getTitle() const;

const std::string
getCopyright() const;

const unsigned
getSvnRevision() const;

const std::string
getGccVersion() const;

const std::string
getGlpkVersion() const;

const std::string
getBoostVersion() const;

const std::string
getOsInfo() const;

const std::string      // say C, POSIX, en_US.iso885915, de_DE.utf8v, see $ locale -a
getLocale() const;

const bool
getReleaseStatus() const;

const std::string
showSplash() const;

const std::string
showInfo
(const xeona::SimKind simKind,
 const unsigned       beepMode,
 const unsigned       exitTrip) const;

const std::string
showHelp
(std::string desc) const;

const std::string
showLegalMessage() const;

const std::string
showXemRules() const;

const std::string
showParse() const;

const std::string
showFinal
(xeona::SimRet simRet,
 const int     exitStatus) const;

const std::string
getLogname() const;

const std::string
getCurrentDate() const;

const std::string
getCurrentTime() const;

const std::string
getReportLevel() const;

const std::string                          // will be empty if not set
getCommandLineModelName() const;

// MANIPULATORS
```

```
  void
  setCommandLineReportLevel
  (const unsigned reportLevel);

  void
  setCommandLineModelName
  (const std::string& modelName);

  // UTILITY FUNCTIONS

private:

  const boost::filesystem::path
  pathFromString
  (const std::string& path) const;           // takes char* too without problem

  const std::string
  extractLeaf                                // the leaf is the final part of the path
  (const boost::filesystem::path path) const;

  const std::string
  extractPath
  (const boost::filesystem::path path) const;

  bool                                       // fail also sets field to ""
  splitString                                // split string and give nominated field
  (const std::string& record,                // input space-separated string
   std::string&       field,                 // selected output field
   const unsigned int index) const;          // field index, zero-based

  std::string
  zeroPadMe
  (const std::string& number,
   const int          padlevel) const;

  std::string
  argProcess
  (int    argc,                              // from main()
   char** argv) const;                       // from main()

  // PRIVATE DATA

private:

  boost::posix_time::ptime         d_tic;         // start time_point
  const boost::filesystem::path    d_progPath;    // complete (absolute) program path
  const boost::filesystem::path    d_startPath;   // complete (absolute) start directory
  std::string                      d_fullCmdLine; // "as typed" less extraneous whitespace
  unsigned                         d_clReportLevel;
  std::string                      d_clModelName;

  // STATIC DATA

private:

  static logga::spLogger s_logger;           // shared_ptr to single logger object

};

#endif // _APPINFO_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : app_except.h
//  file-create-date : Tue 28-Apr-2009 10:14 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : application exception classes / header
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exapp.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This unit implements a hierarchy of custom exception classes
//  for application use.  This hierarchy does not derive from
//  'std::exception'.

//  HEADER GUARD

#ifndef _EXAPP_H_
#define _EXAPP_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/exbase.h"      // xeona exception base class

#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings

//  CAUTION: the '<stdexcept>' header is not needed here but is
//  useful in client code which employs exception specifications
//  in member function declarations: type func(..) throw(..);

#include <stdexcept>          // standard exception classes, runtime_error()

//  CODE

// ----------------------------------------------------------
//  documentation   : Exception classes
// ----------------------------------------------------------
//
//  Entity authors
//
//      Authors coding concrete entities should use class
//      'ent_exception' or (better) a dedicated specialization of
//      this class.  These new specialization can be added to (a
//      possibly new) unit 'ent_except' and not this unit
//      'app_except'.
//
//  New 'xeona::reason' exception file modifications
//
//      common.{h,cc}   : new 'xeona::exit_reason' exit status
```

```
//      a/exapp.{h,cc}  : new 'xeona::reason' exception
//      a/exitstat.cc   : new 'add' call
//      z/client.{h,cc} : add exception specification and, of course, throw call
//
//  Design considerations
//
//      The following authors were consulted on the design of
//      such classes: Lischner (2003), Loudon (2003), Stephens
//      etal (2006), Stroustrup (1997), and Sutter and
//      Alexandrescu (2005).
//
//      The only advice not taken was that from Sutter and
//      Alexandrescu who suggested (p56) it was "preferable" to
//      virtually derive from 'std::exception'.  The inheritance
//      tree used here is stand alone.  And, on reflection, there
//      are clear benefits in keeping STL and xeona exceptions
//      separate.
//
//      Exception classes need to supply:
//
//          - a public destructor
//          - a no-fail constructor
//          - a no-fail copy constructor
//
//      The design here allows for custom information to be
//      embedded, but no logging calls occur.
//
//  Class hierarchy (in namespace 'xeona')
//
//      exception (abstract)            base class
//        + app_exception (abstract)    application use intermediary
//            + kill_on_log             exit if "exit on kill log" set
//            + non_registration        unregistered entity name requested
//            + empty_wrap              misalignment of model data and entity code
//            + short_timeseries        short timeseries
//        + ent_exception (abstract)    for use by entity authors intermediary
//            + entity_issue            general use
//            + ...
//
//  Compiler-supplied copy constructor
//
//      Note the "exception(const exception& orig)" copy
//      constructor supplied by the compiler is okay.
//
//  String formatting conventions
//
//      The formatting of 'd_stringExpl' should add a trailing
//      new line.  The member function 'expl' simply returns this
//      string.
//
//      The formatting of 'd_stringTell' should not add a
//      trailing new line.  The member function 'tell' simply
//      returns this string.
//
//  Throw call example
//
//      throw ( xeona::short_timeseries(getName(), delta) );
//
//  Catch block example
//
//      // within 'main' function
//
//      catch ( const xeona::app_exception& x )
//      {
//        std::ostringstream put;
//        put << x.expl();
//        logger->putx(logga::warn, put);
//        logger->flush();
//
//        std::cout << "  " << x.tell() << " caught"
//                  << ", execution abandoned" << std::endl;
//        std::cout << std::flush;
//
//        return x.code();
//      }
//
// ------------------------------------------------------------

namespace xeona
{
```

```
// ------------------------------------------------------
//  CLASS           : xeona::app_exception (abstract)
// ------------------------------------------------------
//  Description  : abstract base class for application exceptions
//  Role         : use in application code rather than concrete entities
//  Techniques   : exception class, inheritance tree
//  Status       : complete
// ------------------------------------------------------

class app_exception :
  public exception
{
private:

  app_exception();
  app_exception& operator= (const app_exception& orig);   // copy assignment operator

public:

  app_exception
  (const int exitcode);

  virtual ~app_exception() = 0;            // create abstract class

};

// ------------------------------------------------------
//  CLASS           : xeona::kill_on_log
// ------------------------------------------------------
//  Description  : exit on kill "exception" (see design notes)
//  Role         : 'a/logger.cc' and 'Logger::endOfLogCall (const logga::Rank)'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Design notes
//
//      This usage could be considered an abuse of the
//      exception handling system.
//
//      This exception is hopefully but not necessarily caught
//      by 'main' because loggers can reside in global objects.
//      But, at the time of writing (r2500), that is not the
//      case.
//
//  Actual function
//
//      void
//      Logger::endOfLogCall
//      (const logga::Rank rank)
//
// ------------------------------------------------------

class kill_on_log :
  public app_exception
{
private:

  kill_on_log& operator= (const kill_on_log& orig);  // copy assignment oper

public:

  kill_on_log();

};

// ------------------------------------------------------
//  CLASS           : xeona::non_registration
// ------------------------------------------------------
//  Description  : non-registration exception
//  Role         : 'c/factory.cc' and 'EntityFactory::createEntityBind'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Actual function
//
//      shared_ptr<Entity>
//      EntityFactory::createEntityBind
//      (const std::string  entityRegn,
//       const std::string& entityId,
//       Record&            r)
//
```

```
//  Known data
//
//       Identifier, requested class (but read client code for
//       discussion of issues), and entity registration known.
//
// -------------------------------------------------------

class non_registration :
  public app_exception
{
private:

  non_registration();                                          // zero-argument ctor
  non_registration& operator= (const non_registration& orig);  // copy assignment oper

public:

  non_registration
  (const std::string entityIdentifier,     // 'entityId'
   const std::string entityRegistration,   // 'entityRegn'
   const std::string requestedClass);      // 'r.locateClass()'

};

// -------------------------------------------------------
//  CLASS           : xeona::empty_wrap
// -------------------------------------------------------
//  Description  : empty wrap exception
//  Role         : 'c/recset.h' and 'wrap<T>::wrapExit'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Actual function
//
//      template <typename T>
//      void
//      wrap<T>::wrapExit()
//
//  Known data
//
//      Type known.  And the field name would have been logged
//      just prior.
//
// -------------------------------------------------------

class empty_wrap :
  public app_exception
{
private:

  empty_wrap();                                        // zero-argument ctor
  empty_wrap& operator= (const empty_wrap& orig);    // copy assignment operator

public:

  empty_wrap
  (const std::string type);               // 'xeona::demangle(typeid(T).name())'

};

// -------------------------------------------------------
//  CLASS           : xeona::short_timeseries
// -------------------------------------------------------
//  Description  : short timeseries exception
//  Role         : 'c/recset.cc' and 'Field::splitRawStr'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Actual function
//
//      void
//      Field::splitRawStr()
//
//  Known data
//
//      Field name and shortfall known.
//
// -------------------------------------------------------

class short_timeseries :
  public app_exception
```

```
{
private:

  short_timeseries();                                          // zero-argument ctor
  short_timeseries& operator= (const short_timeseries& orig);  // copy assignment oper

public:

  short_timeseries
  (const std::string fieldName,            // 'getName()'
   const int         shortfall);           // 'int delta = horizon - len'

};

// ----------------------------------------------------------
//  CLASS           : xeona::file_not_found
// ----------------------------------------------------------
//  Description  : file not found exception
//  Role         : various, including 'main.cc' and 'main'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Known data
//
//      File name, both leaf and absolute.
//
// ----------------------------------------------------------

class file_not_found :
  public app_exception
{
private:

  file_not_found();                                            // zero-argument ctor
  file_not_found& operator= (const file_not_found& orig);      // copy assignment oper

public:

  file_not_found
  (const std::string filename,
   const std::string comment);

};

// ----------------------------------------------------------
//  CLASS           : xeona::xem_data_issue
// ----------------------------------------------------------
//  Description  : xem issue
//  Role         : used in unit 'recset'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Known data
//
//      In most cases, the "type" and identifier.
//
// ----------------------------------------------------------

class xem_data_issue :
  public app_exception
{
private:

  xem_data_issue();                                            // zero-argument ctor
  xem_data_issue& operator= (const xem_data_issue& orig);      // copy assignment oper

public:

  xem_data_issue
  (const std::string comment,
   const std::string name,
   const std::string valueStr);

};

// ----------------------------------------------------------
//  CLASS           : xeona::lazy_link_fail
// ----------------------------------------------------------
//  Description  : lazy link issue
//  Role         : used in unit 'recset'
//  Techniques   : exception class, inheritance tree (see also 'full_link_fail')
```

```
//  Status       : complete
//
//  Known data
//
//      The sought linkname.
//
// ------------------------------------------------------

class lazy_link_fail :
  public app_exception
{
private:

  lazy_link_fail();                                          // zero-argument ctor
  lazy_link_fail& operator= (const lazy_link_fail& orig);    // copy assignment oper

public:

  lazy_link_fail
  (const std::string comment,
   const std::string linkname);

};

// ------------------------------------------------------
//  CLASS           : xeona::cannot_run_guard
// ------------------------------------------------------
//  Description  : cannot run guard file exception
//  Role         : use in 'main.cc' and 'main'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Known data
//
//      File name.
//
// ------------------------------------------------------

class cannot_run_guard :
  public app_exception
{
private:

  cannot_run_guard();                                          // zero-argument ctor
  cannot_run_guard& operator= (const cannot_run_guard& orig);  // copy assignment oper

public:

  cannot_run_guard
  (const std::string filename);

};

// ------------------------------------------------------
//  CLASS           : xeona::bad_authorship
// ------------------------------------------------------
//  Description  : application-centric tests on badly authored entities
//  Role         : use in 'DomainController::establishDomain'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Known data
//
//      Number of failed entities.
//
// ------------------------------------------------------

class bad_authorship :
  public app_exception
{
private:

  bad_authorship();                                          // zero-argument ctor
  bad_authorship& operator= (const bad_authorship& orig);    // copy assignment oper

public:

  bad_authorship
  (const int numberFailedEntities);

};
```

```
// ----------------------------------------------------------
//  CLASS           : xeona::full_link_fail
// ----------------------------------------------------------
//  Description  : link entity (information flows) failure
//  Role         : use in 'Entity::retFull'
//  Techniques   : exception class, inheritance tree (see also 'lazy_link_fail')
//  Status       : complete
//
//  Known data (perhaps)
//
//      Identifier, cast type, my type (with full/link information).
//
// ----------------------------------------------------------

class full_link_fail :
  public app_exception
{
private:

  full_link_fail();                                          // zero-argument constructor
  full_link_fail& operator= (const full_link_fail& orig);  // copy assignment operator

public:

  full_link_fail
  (const std::string identifier,
   const std::string etype,
   const std::string mtype);

};

// ----------------------------------------------------------
//  CLASS           : xeona::empty_field_on_write
// ----------------------------------------------------------
//  Description  : 'Field::d_timeseries' remains default constructed on write out
//  Role         : use in 'Field::getTimeseries' (zero argument variant)
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Known data
//
//      Single/timeseries, field name.
//
// ----------------------------------------------------------

class empty_field_on_write :
  public app_exception
{
private:

  empty_field_on_write();                                                // zero-arg ctor
  empty_field_on_write& operator= (const empty_field_on_write& orig); // copy assgn opor

public:

  empty_field_on_write
  (const std::string fieldType,            // "single" or "timeseries"
   const std::string fieldname);

};

// ----------------------------------------------------------
//  CLASS           : xeona::timeseries_not_found
// ----------------------------------------------------------
//  Description  : timeseries field is missing
//  Role         : use in 'Field::tieTimeseries'
//  Techniques   : exception class, inheritance tree
//  Status       : complete
//
//  Known data
//
//      Sought field name, template type.
//
// ----------------------------------------------------------

class timeseries_not_found :
  public app_exception
{
private:
```

```
    timeseries_not_found();                                                 // zero-arg ctor
    timeseries_not_found& operator= (const timeseries_not_found& orig); // copy assgn opor

  public:

    timeseries_not_found
    (const std::string fieldname,            // "timeseries"
     const std::string templateType);

  };

  // ------------------------------------------------------------
  //  CLASS           : xeona::yeek_abandon
  // ------------------------------------------------------------
  //  Description  : yeek abandon
  //  Role         : debugging
  //  Techniques   : exception class, inheritance tree
  //  Status       : complete
  //
  //  Known data
  //
  //      Value of 'xeona::yeek'.
  //
  // ------------------------------------------------------------

  class yeek_abandon :
    public app_exception
  {
  private:

    yeek_abandon();                                          // zero-arg ctor
    yeek_abandon& operator= (const yeek_abandon& orig);     // copy assgn opor

  public:

    yeek_abandon
    (const std::string explanation);         // as required

  };

} // namespace 'xeona'

#endif // _EXAPP_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : exbase.h
//  file-create-date : Wed 24-Jun-2009 05:49 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : xeona exception base class / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exbase.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  See file 'a/exapp.h' for overview documentation.

//  HEADER GUARD

#ifndef _EXBASE_H_
#define _EXBASE_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

namespace xeona
{
  // ------------------------------------------------------------
  //  CLASS           : xeona::exception (abstract base)
  // ------------------------------------------------------------
  //  Description  : abstract base class for exceptions
  //  Role         : codebase-wide exception handling
  //  Techniques   : exception class, inheritance tree
  //  Status       : complete
  // ------------------------------------------------------------

  class exception
  {
  private:

    exception();
    exception& operator= (const exception& orig);      // copy assignment operator

  public:

    exception
    (const int exitcode);

    virtual ~exception() = 0;               // create abstract class
```

```
    const std::string expl() const;
    const std::string tell() const;
    const int         code() const;

  protected:

    const int      d_code;              // hardcoded in constructors
    std::string    d_stringExpl;        // formatted on construction, returned by 'expl'
    std::string    d_stringTell;        // formatted on construction, returned by 'tell'

  };

} // namespace 'xeona'

#endif // _EXBASE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : app_except.h
//  file-create-date : Tue 28-Apr-2009 10:14 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : entity exception classes / header
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exent.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This unit implements a hierarchy of custom exception classes
//  for application use.  This hierarchy does not derive from
//  'std::exception'.

//  HEADER GUARD

#ifndef _EXENT_H_
#define _EXENT_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/exbase.h"      // xeona exception base class

#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings

//  CAUTION: the '<stdexcept>' header is not needed here but is
//  useful in client code which employs exception specifications
//  in member function declarations: type func(..) throw(..);

#include <stdexcept>          // standard exception classes, runtime_error()

//  CODE

namespace xeona
{
  // ---------------------------------------------------------
  //  CLASS           : xeona::ent_exception (abstract)
  // ---------------------------------------------------------
  //  Description  : abstract base class for entity exceptions
  //  Role         : use by entity authors within concrete entities
  //  Techniques   : exception class, inheritance tree
  //  Note         : can (and probably should) be further specialized
  //  Status       : complete
  //
  //  Design notes
  //
  //      Users will need to include "a/ent_except.h" in their
  //      unit header and add 'a/ent_except.cc' to their unit
```

```
  //      dependencies in 'machunit'.
  //
  // ----------------------------------------------------------

  class ent_exception :
    public exception
  {
  private:

    ent_exception& operator= (const ent_exception& orig);   // copy assignment operator

  public:

    ent_exception();

    virtual ~ent_exception() = 0;            // create abstract class

  };

  // ----------------------------------------------------------
  //  CLASS           : xeona::entity_issue
  // ----------------------------------------------------------
  //  Description  : generalized entity exception
  //  Role         : use by entity authors within concrete entities
  //  Techniques   : exception class, inheritance tree
  //  Status       : complete
  // ----------------------------------------------------------

  class entity_issue :
    public ent_exception
  {
  private:

    entity_issue();                                         // zero-argument ctor
    entity_issue& operator= (const entity_issue& orig);     // copy assignment operator

  public:

    entity_issue
    (const std::string msgExpl);   // should be suitably formatted with trailing newline

  };

} // namespace 'xeona'

#endif // _EXENT_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : exitstat.h
//  file-create-date : Thu 14-May-2009 07:11 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : exit status database / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exitstat.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _EXITSTAT_H_
#define _EXITSTAT_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <map>                // STL associative container
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : ExitStatus
// ----------------------------------------------------------
//  Description  : stand-alone class holding application return interpretations
//  Role         : used by option '--output'
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class ExitStatus
{
  // TYPEDEFS

private:

  typedef std::map<int, std::string> database_type;

  // DISABLED

private:

  ExitStatus(const ExitStatus& orig);                  // copy constructor
  ExitStatus& operator= (const ExitStatus& orig);      // copy assignment operator

  // CREATORS
```

```
public:

  ExitStatus
  (const std::string notPresentMessage = "(not supported)");

  ~ExitStatus();

  // ACCESSORS

  bool                                      // return 'false' if not present
  operator()
  (const int    exitStatus,
   std::string& exitInterpretation) const;

  std::string
  notPresentMessage() const;

  int
  size() const;

  // UTILITY FUNCTIONS

private:

  void
  add
  (const int         exitStatus,
   const std::string exitInterpretation);

  // INSTANCE DATA

private:

  database_type              d_data;
  const std::string          d_notPresentMessage;

  // INSTANCE DATA

private:

  static logga::spLogger     s_logger;            // shared_ptr to single logger object

};

#endif // _EXITSTAT_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : floatstat.h
//  file-create-date : Mon 20-Jul-2009 10:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : floating-point environment management / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/floatstat.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _FLOATSTAT_H_
#define _FLOATSTAT_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

#include <fenv.h>             // C-style C99 and TR1 floating point environment

//  CODE

// ----------------------------------------------------------
//  CLASS           : xeona::SaveClearResetFeFlag
// ----------------------------------------------------------
//  Description  : save, clear, restore floating-point status flags
//  Role         : floating-point numerics
//  Techniques   : block scope, <fenv.h> functionality as per TR1 (also C99)
//  Status       : complete
//
//  Design notes
//
//      Requires header <fenv.h> (which may migrate to <cfenv>).
//
//      The design is based on code in Becker (2007 pp250-251) --
//      see also (p255) and (p575-576).  Becker is also the best
//      background reference on this subject.
//
//      The class was originally coded with a default constructor
//      argument of 'FE_ALL_EXCEPT', but this was removed because
//      it could lead to somewhat confusing client-side code.
//
//      Note that floating-point exceptions are completely
//      different from C++ exceptions.
//
//  Usage
//
//      Requires block scope, often, but not exclusively, a
```

```
//      function definition:
//
//          {
//            xeona::SaveClearResetFeFlag ouflow(FE_OVERFLOW | FE_UNDERFLOW);
//            // numerical code
//            if ( ouflow.tripped() )
//              // problem resolving code
//            else
//              // okay code
//          }                           // 'status' is destructed on block exit
//
//  CAUTION: 'valgrind' ignores floating-point status flags
//
//      The 'valgrind' memory checker seems to treat all
//      floating-point status flags as unset.  For instance, even
//      the most basic 'valgrind' call will behave as if no
//      floating-point status flags have been set.
//
//          $ ./program                         # floating-point exception tests honored
//          $ valgrind --tool=none ./program    # floating-point exception tests fail
//
//  References
//
//      Becker, Pete.  2007.  The C++ Standard Library extensions :
//        a tutorial and reference.  Addison-Wesley, Upper Saddle
//        River, New Jersey, USA.  ISBN 0-321-41299-0.
//
// ------------------------------------------------------------

namespace xeona
{
  class SaveClearResetFeFlag
  {
    // DISABLED

  private:

    SaveClearResetFeFlag();                                                 // zero-arg ctor
    SaveClearResetFeFlag(const SaveClearResetFeFlag& orig);                 // copy ctor
    SaveClearResetFeFlag& operator= (const SaveClearResetFeFlag& orig);  // copy assn oper

    // CREATORS

  public:

    SaveClearResetFeFlag
    (const int feFlag);                     // macros as defined in <fenv.h>, can be |'ed

    ~SaveClearResetFeFlag();

    // ACCESSORS

  public:

    bool                                    // 'true' if flag has been reset
    tripped() const;

    int                                     // return constructor argument
    getFlag() const;

    // INSTANCE DATA

  private:

    const int    d_feFlag;                  // refer <fenv.h>
    fexcept_t    d_feStat;                  // type representing floating-point status
  };

} // namespace 'xeona'

#endif // _FLOATSTAT_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : helpers.h
//  file-create-date : Mon 18-May-2009 11:08 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : application helper functions / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/helpers.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This unit contains functions and classes for use by the
//  application but which don't sensibly belong elsewhere.

//  HEADER GUARD

#ifndef _HELPERS_H_
#define _HELPERS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ---------------------------------------------------------
//  FREE FUNCTION   : xeona::logRankToGlpkLevel
// ---------------------------------------------------------
//  Description  : logging rank (via '--report' option) to solver noise policy
//  Role         : optional use prior to creating new 'SolverIf' objects
//  Techniques   : switch statement
//  Status       : complete but not tested
//
//  Design notes
//
//      'svif::ReportingLevel' enum members
//      (taken from unit 'siglp' documentation)
//
//          not_specified = 0
//          silent        = 1    GLPK no output (although leakage may occur)
//          low           = 2    GLPK errors and warning
//          medium        = 3    GLPK normal             + problem info
//          high          = 4    GLPK normal plus info   + problem info plus GLPK report
//
//      These values equate to ONE more than the new GLPK
//      'msg_lev' integers GLP_MSG_{OFF,ERR,ON,ALL} (and the
//      undocumented GLP_MSG_DBG = 4) and the old 'LPX_K_MSGLEV'
```

```
//      parameter values.
//
//
//  CAUTION: required header
//
//      "d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
//
// -------------------------------------------------------

namespace xeona
{
  const svif::ReportingLevel
  logRankToGlpkLevel
  (const unsigned loggerRank);               // fundamentally 'logga::Rank'

} // namespace 'xeona'

#endif // _HELPERS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : logger.h
//  file-create-date : Thu 05-Apr-2007 11:33 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : run-time logging functionality / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/logger.h $

//  HEADER GUARD

#ifndef _LOGGER_H_
#define _LOGGER_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/exapp.h"       // application exception classes
#include "../c/smart_ptr.h"   // switch easily between TR1 and Boost smart pointers

#include <map>                // STL associative container
#include <ostream>            // output streams
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <boost/date_time/posix_time/posix_time.hpp>  // plus io
#include <boost/algorithm/string_regex.hpp>           // additional regex support

// CAUTION: more system includes than would be usual are required
// because the 'repx' function templates are also defined in this
// file

#include <iomanip>            // setw() and family
#include <iostream>           // standard io

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions
#include <boost/regex.hpp>              // regular expression support

//  CODE

// ------------------------------------------------------
//  notes           : TR1 or Boost shared pointers
// ------------------------------------------------------
//
//  Until and including r2211, this unit was hard coded for
//  'std::tr1::shared_ptr'.  From r2212, the local 'smart_ptr.h'
//  header was included, which controls whether TR1 or Boost code
//  is used, and the declarations here duly modified.
//
//  For other applications, this unit can be reverted to the
//  original TR1 code.
//
// ------------------------------------------------------
```

```
//  notes           : Implementation code in header
// ------------------------------------------------------
//
//  Some of the implementation code is placed in this header.
//
//  The reason is that the final lines of this file contain two
//  preprocessor hash-define macros which act on the member
//  functions 'repx' and 'putx'.  If enabled, these macros make
//  __FILE__, __LINE__, and __func__ information available these
//  functions and thus enable this information to be logged to
//  the selected output stream.
//
//  Moreover the 'repx' function is also templated and a full
//  list of instantiations cannot be fully identified in advance.
//  This fact alone would make this function a candidate for
//  definition in the header file.
//
//  Note also that the two member functions and each variant
//  (four functions in total) need to 'inlined' in order to avoid
//  a link-time "multiple definition error".

namespace logga
{
  // ------------------------------------------------------
  //  TYPEDEF         : logga::spLogger
  // ------------------------------------------------------
  //  Purpose       : simply for coding convenience
  // ------------------------------------------------------

  class Logger;                                 // forward declaration
  typedef shared_ptr<logga::Logger> spLogger;

  // ------------------------------------------------------
  //  ENUM            : logga::Rank
  // ------------------------------------------------------
  //  Purpose       : set of enums to influence reporting
  //  Application   : in function arguments
  //  Status        : complete
  // ------------------------------------------------------

  // CAUTION: these enums are not proceeded by "e_" in the interests of brevity

  enum Rank
    {
      yeek = 0,                                 // for use during development only
      kill = 1,
      warn = 2,
      info = 3,
      dbug = 4,
      xtra = 5,                                 // extra reporting, say from destructors
      adhc = 6                                  // ad-hoc and insignificant reporting
    };

  // ------------------------------------------------------
  //  VARIABLE        : logga::rankNoData
  //  VARIABLE        : logga::rankJumpy
  // ------------------------------------------------------

  extern Rank rankNoData;                       // set by '--nodata'
  extern Rank rankJumpy;                        // set by '--jumpy'

  // ------------------------------------------------------
  //  CLASS           : logga::Logger
  // ------------------------------------------------------
  //  Purpose       : universal logging object for each 'ostream'
  //  Status        : complete
  //  Comment       : the formatting code might be better expressed using Boost.Format
  //
  //  CAUTION: unusual code placement
  //
  //      Certain function are defined in this header file due to
  //      the preprocessor macro.  See elsewhere for details.
  //
  //  CAUTION: use of 'repx' in 'const' member functions
  //
  //     Calls to 'this->repx(..)' cannot be used in member
  //     functions deemed 'const' because 'repx' does not embed
  //     this guarantee.
  //
  // ------------------------------------------------------
```

```
class Logger
{
  // PRIVATE ENUMS

private:

  enum CallType                             // used to control additional blank lines
    {
      e_none     = 0,                       // as an initial value
      e_repx     = 1,                       // previous call was 'repx'
      e_putx     = 2,                       // previous call was 'putx'
      e_addblank = 3                        // add blank line before next logging output
    };

  // DISABLED

private:

  Logger();                                 // prevent instantiation without an ostream
  Logger(const Logger& orig);               // copy constructor
  Logger& operator= (const Logger& orig);   // copy assignment operator

  // CREATORS

public:

  Logger
  (std::ostream& os);

  ~Logger();

  // the 'repx' suite of functions are suitable for
  // preprocessor overwrite, in which case the file/line/func
  // versions are called

  // LOGGING FUNCTIONS

  template <typename T>
  inline
  bool                                      // 'true' means message was displayed
  repx                                      // normal single line logging
  (const logga::Rank rank,                  // the "logga::" is, in fact, not needed
   const std::string text,                  // user supplied text
   const T&          value);                // implicit instantiation used

  template <typename T>
  inline
  bool
  repx
  (const logga::Rank rank,
   const std::string text,                  // user supplied text
   const T&          value,                 // implicit instantiation used
   const std::string file,                  // from preprocessor macro __FILE__
   const int         line,                  // from preprocessor macro __LINE__
   const std::string func);                 // from preprocessor macro __func__

  bool
  putx                                      // string-stream multi-line logging
  (const logga::Rank   rank,
   std::ostringstream& oss);                // users string-stream which is also nulled

  bool
  putx
  (const logga::Rank   rank,
   std::ostringstream& oss,                 // users string-stream which is also nulled
   const std::string   file,                // from preprocessor macro __FILE__
   const int           line,                // from preprocessor macro __LINE__
   const std::string   func);               // from preprocessor macro __func__

  bool
  dotx                                      // output a row of dots or similar
  (const logga::Rank  rank,
   const std::string& text);

  bool
  dotx                                      // output a row of dots or similar
  (const logga::Rank  rank,
   const std::string& text,
   const std::string  file,                 // from preprocessor macro __FILE__
   const int          line,                 // from preprocessor macro __LINE__
   const std::string  func);                // from preprocessor macro __func__
```

```
    bool
    test                                    // insert a TEST COUNT COMMENCING line
    (const int testCount);                  // 'testCount' is the desired test count

    bool
    test
    (const int         testCount,
     const std::string putMsg);             // 'put'-type message without "\n"

    void
    addSmartBlank();                        // add blank line in a relatively smart way

    void
    addSmartBlank                           // ditto, but only if rank-appropriate
    (const logga::Rank rank);

    void
    addDumbBlank();                         // add blank line without question

    void
    addDumbBlank                            // ditto, but only if rank-appropriate
    (const logga::Rank rank);

    void
    addFinalStdoutBlank();

    void
    flush();                                // explicitly flush the output buffer,
                                            //   but with no other action
    void
    beep                                    // using system utility or bell character
    (int type);                             // controls type of beep

    // ACCESSORS

    unsigned getYeekAllCount() const { return d_yeekAllCount; }
    unsigned getKillAllCount() const { return d_killAllCount; }
    unsigned getWarnAllCount() const { return d_warnAllCount; }
    unsigned getInfoAllCount() const { return d_infoAllCount; }
    unsigned getDbugAllCount() const { return d_dbugAllCount; }
    unsigned getXtraAllCount() const { return d_xtraAllCount; }
    unsigned getAdhcAllCount() const { return d_adhcAllCount; }

    unsigned getYeekFixedCount() const { return d_yeekFixedCount; }
    unsigned getKillFixedCount() const { return d_killFixedCount; }
    unsigned getWarnFixedCount() const { return d_warnFixedCount; }
    unsigned getInfoFixedCount() const { return d_infoFixedCount; }
    unsigned getDbugFixedCount() const { return d_dbugFixedCount; }
    unsigned getXtraFixedCount() const { return d_xtraFixedCount; }
    unsigned getAdhcFixedCount() const { return d_adhcFixedCount; }

    unsigned getYeekResetCount() const { return d_yeekResetCount; }
    unsigned getKillResetCount() const { return d_killResetCount; }
    unsigned getWarnResetCount() const { return d_warnResetCount; }
    unsigned getInfoResetCount() const { return d_infoResetCount; }
    unsigned getDbugResetCount() const { return d_dbugResetCount; }
    unsigned getXtraResetCount() const { return d_xtraResetCount; }
    unsigned getAdhcResetCount() const { return d_adhcResetCount; }

    logga::Rank
    getReportLevelRank() const;

    unsigned
    getReportLevelInt() const;

    std::string
    getReportLevelStr() const;

    const std::string                       // all space-separated ranks
    getAllTriggerStr() const;

    const std::string                       // has an uncharacteristic trailing newline
    summarizeRankCounts() const;            // makes small table

    const std::string
    getTrigger() const;                     // as set by '--watch' option

    // STATIC MANIPULATORS

    static
```

```
    bool
    enableConsoleTruncation();              // on by default, set using '_XCOLS' macro

    static
    bool
    disableConsoleTruncation();             // do not truncate output to suit '_XCOLS'

    // MANIPULATORS

    bool
    enableBeeping();                        // general activation of beeping

    bool
    disableBeeping();                       // general suppression of beeping

    void
    setBeepOnCompletion();                  // beep call made from '~Logger'

    void
    setBeepOnOrAbove                        // run-time beep trigger
    (logga::Rank rank);

    void
    setTrigger                              // as set by --trigger option
    (const std::string& trigger);

    void
    resetRankCounts                         // at given level and noisier
    (const unsigned reportLevel = 1);       // defaults to logga::kill

    Rank                                    // return previous report level
    setReportLevel                          // also resets triggers based on reportLevel
    (const unsigned reportLevel);           // will also accept a logga::Rank

    bool                                    // 'false' indicates 'oss.good()' failed
    resetOSS                                // restore string-stream after manip'ing
    (std::ostringstream& oss);              // user-supplied string-stream to be reset

    void
    updateReturnStatus
    (const int         returnStatus,
     const std::string returnInterpetation = "(not forwarded)");

    // UTILITY FUNCTIONS

  private:

    bool
    shouldLog
    (const std::string& file);

    void
    firstCall();                            // print space, header, and rule

    void
    printHeader();                          // print reporting header

    void
    printRule                               // print reporting rule
    (const int skip);                       // number of blank lines to follow rule

    void
    incrementRankCounts                     // update counters
    (unsigned rank);                        // will also accept a logga::Rank

    std::string
    formatTriggerStr                        // used by 'getAllTriggerStr'
    (logga::Rank rank,
     unsigned    count) const;

    void
    formatRankCount                         // used by 'summarizeRankCounts'
    (std::ostringstream& os,                // passed-in string-stream
     const std::string&  term,
     unsigned            all,
     unsigned            fixed,
     unsigned            reset) const;

    void
    endOfLogCall                            // only reacts to first kill
    (const logga::Rank rank)
```

```
     throw(std::exception,                  // exception specification
           xeona::kill_on_log);

   std::string                              // current timestamp, truncated to 0.00 secs
   timestampUTC();

   std::string               // zero-padded truncated number, say, 5.003699 -> 05.0037
   numToPad                                 // used in 'calcInterval'
   (double input,
    int    LEFT  = 2,                       // zero-padded digits left of decimal point
    int    RIGHT = 4);                      // digits right of decimal point

   std::string                              // return was-now interval in forms 00.0000s,
   calcInterval();                          //   00.0000m, or 00.0000h, or complain

   std::string                              // four character string describing rank
   calcRank                                 // obtain rank term for use in reporting
   (const Rank  rank) const;

   // INSTANCE DATA

 private:

   std::ostream&               d_os;            // can be either console or file ostream
   boost::posix_time::ptime    d_was;           // point in time object
   CallType                    d_lastCall;      // track last call type
   unsigned                    d_repCount;      // report call count
   Rank                        d_reportLevel;   // can be reset on command-line
   bool                        d_enableBeep;    // controls beeping
   bool                        d_beepCompletion; // defines beeping behavior
   Rank                        d_beepThreshold;  // defines beeping behavior
   std::string                 d_triggerStr;    // string trigger for added reporting
   boost::regex                d_triggerRegex;  // regex trigger for added reporting
   bool                        d_finalBlank;    // add final blank line to stdout

   int                         d_returnStatus;  // supplied at end of main
   std::string                 d_returnInterp;  // supplied at end of main

   unsigned d_yeekNoFirst;                  // close-of-application reporting
   unsigned d_killNoFirst;                  // for recording no (= number) of first call
   unsigned d_warnNoFirst;

   unsigned d_yeekAllCount;                 // close-of-application reporting
   unsigned d_killAllCount;                 // a simple count of calls
   unsigned d_warnAllCount;
   unsigned d_infoAllCount;
   unsigned d_dbugAllCount;
   unsigned d_xtraAllCount;
   unsigned d_adhcAllCount;

   unsigned d_yeekFixedCount;               // close-of-application reporting
   unsigned d_killFixedCount;               // a simple count of prints
   unsigned d_warnFixedCount;
   unsigned d_infoFixedCount;
   unsigned d_dbugFixedCount;
   unsigned d_xtraFixedCount;
   unsigned d_adhcFixedCount;

   unsigned d_yeekResetCount;               // for test purposes
   unsigned d_killResetCount;               // simple count of reset prints
   unsigned d_warnResetCount;
   unsigned d_infoResetCount;
   unsigned d_dbugResetCount;
   unsigned d_xtraResetCount;
   unsigned d_adhcResetCount;

   // STATIC DATA

 private:

   static bool               s_truncate;         // console truncation flag

   // in-class initialization of static constant integers is acceptable

   static const unsigned int s_MAR  =       0;   // left margin
   static const unsigned int s_GUT  =       3;   // standard gutter
   static const unsigned int s_OVR  =       2;   // overlength string separation
   static const unsigned int s_RULE =     130;   // rule length
   static const unsigned int s_TERM = _XTCOLS;   // external macro set via CPPFLAGS

   static const unsigned int s_TAB1 =       4;   // file line number field
```

```
  static const unsigned int s_TAB2 =    12+3;   // file name field, was 12+
  static const unsigned int s_TAB3 =      20;   // function name field
  static const unsigned int s_TAB4 =       4;   // report counter
  static const unsigned int s_TAB5 =       8;   // delta-t field
  static const unsigned int s_TAB6 =       4;   // rank field
  static const unsigned int s_TAB7 =      35;   // message field

};

// ----------------------------------------------------------
//  MEMBER FUNCTION : logga::Logger::repx (without compiler macros)
// ----------------------------------------------------------

template <typename T>
inline
bool
Logger::repx
(const logga::Rank rank,
 const std::string text,
 const T&          value)
{
  // check verbosity
  if ( rank > d_reportLevel )
    {
      incrementRankCounts(rank);           // update rank register
      return false;
    }
  else
    {
      if ( d_repCount == 0 ) firstCall();  // note first call
      incrementRankCounts(rank);           // update rank register
    }

  // local ostringstream buffer
  std::ostringstream ssBuf;

  // use bool (and boost::logic::tribool) terms and not digits
  ssBuf << std::boolalpha;

  // 'setprecision' normally controls the number of digits on
  // both side of the decimal point for floating point numbers,
  // but placed after 'fixed' it controls the number of digits
  // on the right side of the decimal point -- namely:
  // 5500000.00

  ssBuf << std::fixed
        << std::setprecision(2);

  ssBuf << std::setw(s_TAB4 + s_MAR) << std::right << d_repCount
        << std::setw(         s_GUT) << ""
        << std::setw(s_TAB5        ) << std::left  << calcInterval()
        << std::setw(         s_GUT) << ""
        << std::setw(s_TAB6 + s_GUT) << std::left  << calcRank(rank)
        << std::setw(s_TAB7        ) << std::left  << text
        << std::setw(         s_OVR) << ""
        <<                             std::left  << value;

  // the main stringstream is truncated to avoid line wrapping
  // on the console

  std::string strBuf = ssBuf.str();             // stringify
  if ( strBuf.length() > s_TERM && s_truncate)  // truncate if necessary
    {
      strBuf.resize(s_TERM - 2);
      strBuf += " >";                           // truncation symbol
    }
  boost::trim_right(strBuf);                    // trim trailing spaces in-situ

  // FOURTH: unrelated tasks

  // add blank line as needed
  if ( d_lastCall == e_putx )
    d_os << "\n";                          // insert newline
  if ( d_lastCall == e_addblank )
    d_os << "\n";

  // update status
  d_lastCall = e_repx;

  // write out main message
  d_os << strBuf << "\n";
```

```
    ++d_repCount;

    // housekeeping
    endOfLogCall(rank);
    return true;
  }

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : logga::Logger::repx (with compiler macros)
  // ------------------------------------------------------------

  // CAUTION: code quality: this code may be better reworked
  // using Boost.Format library calls for at least some of the
  // functionality

  template <typename T>
  inline
  bool
  Logger::repx
  (const logga::Rank rank,
   const std::string text,
   const T&          value,
   const std::string file,                  // from preprocessor macro __FILE__
   const int         line,                  // from preprocessor macro __LINE__
   const std::string func)                  // from preprocessor macro __func__
  {
    // check verbosity
    if ( rank > d_reportLevel               // 'rank' exceeds 'd_reportLevel' threshold
         && ! shouldLog(file) )
      {
        incrementRankCounts(rank);          // update rank register
        return false;
      }
    else
      {
        if ( d_repCount == 0 ) firstCall();  // note first call
        incrementRankCounts(rank);          // update rank register
      }

    // FIRST: the line (line), file (source), func (call)
    // information is processed and then turned into a local
    // string -- this constrains long filenames by allowing them
    // to push into the call column but no further

    std::ostringstream ssTmp;               // for line, file, and func information
    std::string        strTmp;              // string version of above
    int                tab;                 // setw allowance in main stringstream

    // ALSO: process the 'file' string to remove valuless path information

    std::string path(file);                 // also removes 'const' status

#if 0 // _XRELEASE

    // conditional compilation to shorten link-time for
    // non-release builds -- CAUTION: must be conditional
    // compilation and not an 'xeona::releaseStatus' if block in
    // order to do so

    // EXTREME CAUTION: leave this first macro block disabled --
    // it causes segfaults, perhaps in 'logga::Logger::~Logger',
    // with more complex models like 'submodel.11' under
    // '--report 3' or greater -- see commit r4240 and earlier
    // for more details

    // leftmost longest rule and repetition apply: "a/../b/../c/header.h" -> "c/header.h"

    const std::string rgxstr = "([[:alnum:]]+/\\.\\./)";
    const std::string patstr = "";                  // \\1 is 'rgxstr' match, try "\\1--"
    boost::regex pattern(rgxstr);                   // make a regular expression
    path = boost::regex_replace                     // replace variant
      (path,                                        // so-called target sequence
       pattern,                                     // regular expression
       patstr,                                      // format string
       boost::match_default | boost::format_sed);   // search options, 'sed' is simplest

#else // use a more basic alternative (added later, in case you're wondering)

    while ( path.substr(1, 4) == "/../" )     // assume SINGLE character directories
      {
        path.erase(0, 5);
```

```
      }

#endif // _XRELEASE

    // carry on as before with 'path' instead of 'file'

    ssTmp << std::setw(s_TAB1 + s_MAR) << std::right << line
          << std::setw(         s_GUT) << ""
          << std::setw(s_TAB2        ) << std::left  << path     // was 'file'
          << std::setw(         s_OVR) << ""
          << std::setw(s_TAB3 + s_GUT) << std::left  << func;

    strTmp = ssTmp.str();
    tab    = s_TAB1 + s_MAR + s_GUT + s_TAB2 + s_TAB3 + s_GUT;
    boost::trim_right(strTmp);               // trim trailing spaces in-situ

    // SECOND: the main stringstream is built

    std::ostringstream ssBuf;                // main stringstream

    // use bool (and boost::logic::tribool) terms and not digits
    ssBuf << std::boolalpha;

    // 'setprecision' normally controls the number of digits on
    // both side of the decimal point for floating point numbers,
    // but placed after 'fixed' it controls the number of digits
    // on the right side of the decimal point -- namely:
    // 5500000.00

    ssBuf << std::fixed
          << std::setprecision(2);

    ssBuf << std::setw(tab)             << std::left  << strTmp   // line, file, func
          << std::setw(         s_OVR) << ""
          << std::setw(s_TAB4        ) << std::right << d_repCount
          << std::setw(         s_GUT) << ""
          << std::setw(s_TAB5        ) << std::left  << calcInterval()
          << std::setw(         s_GUT) << ""
          << std::setw(s_TAB6 + s_GUT) << std::left  << calcRank(rank)
          << std::setw(s_TAB7        ) << std::left  << text
          << std::setw(         s_OVR) << ""
          <<                             std::left  << value;

    // THIRD: the main stringstream is truncated to avoid line
    // wrapping on the console

    std::string strBuf = ssBuf.str();             // stringify
    if ( strBuf.length() > s_TERM && s_truncate)  // truncate if necessary
      {
        strBuf.resize(s_TERM - 2);
        strBuf += " >";                           // truncation symbol
      }
    boost::trim_right(strBuf);                    // trim trailing spaces in-situ

    // FOURTH: unrelated tasks

    // add blank line as needed
    if ( d_lastCall == e_putx )
      d_os << "\n";                           // insert newline
    if ( d_lastCall == e_addblank )
      d_os << "\n";

    // update status
    d_lastCall = e_repx;

    // write out main message
    d_os << strBuf << "\n";
    ++d_repCount;

    // housekeeping
    endOfLogCall(rank);
    return true;
  }

  // ----------------------------------------------------------
  //  MEMBER FUNCTION : logga::Logger::putx (with/without compiler macros)
  // ----------------------------------------------------------

  inline                                    // CAUTION: protects against link errors
  bool
  Logger::putx
```

```
(const logga::Rank   rank,
 std::ostringstream& oss)                  // users string-stream which is also nulled
{
  // check if needed
  if ( oss.str().empty() )
    return false;                          // simply quit if 'oss' is empty

  // check verbosity
  if ( rank > d_reportLevel )
    {
      oss.str("");                         // empty the non-const o-string-stream [1]
      return false;
    }

  // [1] emptying an ostringsteam: 'str(const std::string s)'
  // will set the buffer contents to 's', see Lischner (2003
  // p653) and elsewhere

  // add blank line as needed
  if ( d_lastCall == e_repx )
    d_os << "\n";                          // insert newline
  if ( d_lastCall == e_addblank )
    d_os << "\n";

  // update status
  d_lastCall = e_putx;

  // write out main message
  d_os << oss.str();
  oss.str("");                             // empty the non-const o-string-stream [1]

  // housekeeping
  endOfLogCall(rank);
  return true;
}

inline
bool
Logger::putx
(const logga::Rank   rank,
 std::ostringstream& oss,                  // users string-stream which is also nulled
 const std::string  file,                  // from preprocessor macro __FILE__
 const int          line,                  // from preprocessor macro __LINE__
 const std::string  func)                  // from preprocessor macro __func__
{
  // check if needed
  if ( oss.str().empty() )
    return false;                          // simply quit if 'oss' is empty

  // check verbosity
  if ( rank > d_reportLevel                // 'rank' exceeds 'd_reportLevel' threshold
       && ! shouldLog(file) )
    {
      oss.str("");                         // empty the non-const o-string-stream [1]
      return false;
    }

  // [1] emptying an ostringsteam: 'str(const std::string s)'
  // will set the buffer contents to 's', see Lischner (2003
  // p653) and elsewhere

  // add blank line as needed
  if ( d_lastCall == e_repx )
    d_os << "\n";                          // add a newline
  if ( d_lastCall == e_addblank )
    d_os << "\n";

  // update status
  d_lastCall = e_putx;

  // write out main message
  d_os << oss.str();                       // I think this is the best (and only?) way
  oss.str("");                             // empty the non-const o-string-stream [1]

  // housekeeping
  endOfLogCall(rank);
  return true;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : logga::Logger::dotx (with/without compiler macros)
```

```
// ----------------------------------------------------------

inline
bool
Logger::dotx
(const logga::Rank  rank,
 const std::string& text)
{
  // process information
  char c = ' ';
  switch ( rank )
    {
    case yeek: c = '*'; break;
    case kill: c = '-'; break;
    case warn: c = '-'; break;
    case info: c = '-'; break;
    case dbug: c = '-'; break;
    case xtra: c = '-'; break;
    case adhc: c = '-'; break;
    default: std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
    }
  std::ostringstream oss;
  oss << "  "
      << "  " << std::string(2, c)
      << "  " << calcRank(rank)
      << "  " << std::string(20, c)
      << "  " << text
      << "\n";
  return putx(rank, oss);                  // simplifies the coding!
}

inline
bool
Logger::dotx
(const logga::Rank  rank,
 const std::string& text,
 const std::string  file,                  // from preprocessor macro __FILE__
 const int          line,                  // from preprocessor macro __LINE__
 const std::string  func)                  // from preprocessor macro __func__
{
  const unsigned len = 60;                 // hardcoded length of output string

  // process information
  char c = '\0';                           // ASCII 0 null character
  switch ( rank )
    {
    case yeek: c = '*'; break;
    case kill: c = '-'; break;
    case warn: c = '-'; break;
    case info: c = '-'; break;
    case dbug: c = '-'; break;
    case xtra: c = '-'; break;
    case adhc: c = '-'; break;
    default: std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
    }

  const std::string macroSep = "  ";       // reporting separator
  std::ostringstream tmp;
  tmp << " "
      << " " << std::string(2, c)
      << " " << calcRank(rank)
      << " " << std::string(4, c)
      << " " << line << macroSep << file << macroSep << func
      << " " << std::string(len, c);
  std::string str = tmp.str();
  if ( str.length() > len )                // strictly unnecessary
    str.erase(len);                        // truncate
  std::ostringstream oss;
  oss << str << " " << text << "\n";
  return putx(rank, oss);                  // simplifies the coding!
}

// ----------------------------------------------------------
//  FREE FUNCTION   : logga::refLogStream
// ----------------------------------------------------------
//  Status       : obsolete, used ptrLogStream instead
//
//  CAUTION: code placement
//
//      Must be preceded by full Logger declaration.
//
```

```
  //  Note
  //
  //      See implementation (.cc) file for design analysis.
  //
  // ------------------------------------------------------------

  Logger&
  refLogStream();

  // ------------------------------------------------------------
  //  FREE FUNCTION   : logga::ptrLogStream
  // ------------------------------------------------------------
  //  Status       : use in preference to refLogStream
  //
  //  CAUTION: code placement
  //
  //      Must be preceded by full Logger declaration.
  //
  //  Note
  //
  //      See implementation (.cc) file for design analysis.
  //
  // ------------------------------------------------------------

  shared_ptr<Logger>
  ptrLogStream();

  // ------------------------------------------------------------
  //  FREE FUNCTION   : logga::checkReportLevel
  // ------------------------------------------------------------

  bool                                        // 'true' if 'trip' would print
  checkReportLevel
  (const logga::Rank trip);

} // namespace 'logga'

// ------------------------------------------------------------
//  CPP MACROS      : logga::repx, logga::putx
// ------------------------------------------------------------

// Preprocessor macro to gain additional reporting under debug
// compilation, noting that this macro degrades gracefully if the
// hash-define is not made.  The use of brackets around first
// three arguments is also recommended.

#if 1 // 0 is deactivated, 1 is activated
#  define repx(a, b, c) repx((a), (b), (c), __FILE__, __LINE__, __func__)
#  define putx(a, b)    putx((a), (b),      __FILE__, __LINE__, __func__)
#  define dotx(a, b)    dotx((a), (b),      __FILE__, __LINE__, __func__)
#endif

#endif // _LOGGER_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : recorder.h
//  file-create-date : Mon 26-Apr-2010 12:10 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : summarizing recorder class / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/recorder.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Class 'EventRecorder' was originally written to help debug
//  the CTA algorithm.  It can be used more generally however.

//  HEADER GUARD

#ifndef _RECORDER_H_
#define _RECORDER_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro

//  FORWARD (PARTIAL) DECLARATIONS

class DomainController;
class Gateway;

// ----------------------------------------------------------
//  CLASS           : EventRecorder
// ----------------------------------------------------------
//  Description  : event recorder which produces a formatted summary on demand
//  Role         : originally written for tracking the CTA algorithm
//  Techniques   : nested class, 'std::vector', streaming
//  Status       : complete
//
//  Design notes
//
```

```
//      Intended for recording events within the CTA algorithm.
//
//  Sample output
//
//      cta summary
//        step = 10
//        cnt   event             details                                 remark
//        ------------------------------------------------------------------------------
//          0   EventRecorder     event recorder under construction     cta
//          1   cta               CtaSimple
//          2   event             details                               remark
//      *   3   capset            gate-1 (from domcon-1)                capset / reason
//      *   4   note              some long note                        uh-ohh!
//
// -----------------------------------------------------

class EventRecorder
{
  // NESTED CLASSES

private:

  // -----------------------------------------------------
  //  CLASS           : EventRecorder::Event (private)
  // -----------------------------------------------------

  class Event
  {
    // DISABLED

  private:

    Event();                                // zero-argument constructor

    // CREATORS

  public:

    Event
    (const std::string& event,
     const std::string& details,
     const std::string& remark);

    // MANIPULATORS

    void
    addAlert
    (const std::string& remark);

    // ACCESSORS

    std::string
    outputEvent
    (const unsigned count) const;

    // INSTANCE DATA

  private:

    bool           d_alert;
    std::string    d_event;
    std::string    d_details;
    std::string    d_remark;

  };

  // -----------------------------------------------------
  //  CLASS           : EventRecorder (public interface))
  // -----------------------------------------------------

  // DISABLED

private:

  EventRecorder();                                          // zero-argument constructor
  EventRecorder(const EventRecorder& orig);                 // copy constructor
  EventRecorder& operator= (const EventRecorder& orig);     // copy assignment operator

  // CREATORS

public:
```

```
  EventRecorder
  (const std::string title,
   const unsigned    step);

  ~EventRecorder();

  // MANIPULATORS

public:

  void
  event                                     // general reporting
  (const std::string event,
   const std::string details,
   const std::string remark);

  void
  event                                     // general reporting
  (const std::string event,
   const std::string details,
   const double      value);

  void
  event                                     // general reporting
  (const std::string event,
   const std::string details,
   const int         value);

  void
  cta                                       // specialized reporting
  (const std::string klass,                 // for example 'CtaSimple'
   const std::string remark = "");

  void
  mark
  (const std::string            symbol,     // for example, "theta"
   shared_ptr<Gateway>          gate,
   shared_ptr<DomainController> domcon,
   const std::string            remark = "");

  void
  unmark
  (const std::string            symbol,     // for example, "theta"
   shared_ptr<Gateway>          gate,
   shared_ptr<DomainController> domcon,
   const std::string            remark = "");

  void
  hop
  (shared_ptr<Gateway>          gate,
   shared_ptr<DomainController> fromDomcon,
   shared_ptr<DomainController> toDomcon,
   const std::string            remark = "");

  void
  capset                                    // specialized reporting
  (shared_ptr<DomainController> domcon,
   shared_ptr<Gateway>          targetGate,
   const std::string            remark = "");

  void
  transolve                                 // specialized reporting
  (shared_ptr<DomainController> domcon,
   const std::string            remark = "");

  void
  note                                      // general reporting
  (const std::string note);

  void
  alert                                     // add alert information subsequently
  (const std::string remark = "");

  // ACCESSORS

  std::string                               // final newline not required
  output() const;                           // output a summary

  // INSTANCE DATA
```

```
private:

  const std::string     d_title;
  const int             d_step;
  std::vector<Event>    d_events;

};

#endif // _RECORDER_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xedocs.h
//  file-create-date : Tue 23-Sep-2008 06:28 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : processing of 'xedoc' entity documentation / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/xedocs.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//      Brittle behavior: the process of scanning and building
//      xedoc information breaks in the face of multiple blank
//      lines in a xedoc block -- one fix is to modify the
//      'xedox' script to "squeeze" the input, another is to
//      check here that each entry finishes with header
//      information.
//
//  This unit requires that the text file defined by
//  'XEDOCS_FILE' in 'common.cc' be present.  Moreover the
//  contents of this text file (even if lacking information)
//  must:
//
//      * be set in double quotes ("")
//      * comprise a single line
//      * replace genuine newline chars (ASCII 10 for UNIX) with "\n" substrings
//      * replace genuine double quote chars with "\"" substrings
//
//  That said, leading and trailing blank lines are okay.
//
//  Additional requirements for the structuring of data are
//  discussed elsewhere.  But here is a simple example:
//
//      "entity.one\n\n    class  > One\n\n\nentity.two\n\n    class  > Two\n"

//  HEADER GUARD

#ifndef _XEDOCS_H_
#define _XEDOCS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <map>                // STL associative container

//  CODE

// ----------------------------------------------------------
//  CLASS           : Xedocs
// ----------------------------------------------------------
```

```
//  Description  : usually short-lived and single object called from within 'main.cc'
//  Role         : support for the '--class arg' option
//  Techniques   : 'std::map' key/value database, underlying data embedded at compile-time
//  Status       : complete
//
//  Design notes
//
//      The underlying data is held in the string
//      'xeona::xedocsFileContent' from the 'common' unit and is
//      "loaded" at compile-time using a preprocessor
//      hash-include directive.  The hash-included filename is
//      held in the string 'xeona::xedocsFileName', also from the
//      'common' unit.
//
//      The hash-included file is generated (and later nullified)
//      by the 'mach' build script.  Hence, the binary must be
//      built using a call to 'mach' and not 'make'.
//
//      The 'mach' script calls the 'xedoc' script to refresh the
//      said hash-included file.
//
// ----------------------------------------------------------

class Xedocs
{
  // LOCAL ENUMERATIONS

public:

  enum FindStatus
    {
      e_unknown = 0,
      e_classFound,
      e_classNotFound,
      e_emptyDatabase
    };

  // DISABLED

private:

  Xedocs(const Xedocs& orig);                // copy constructor
  Xedocs& operator= (const Xedocs& orig);    // copy assignment operator

  // CREATORS

public:

  Xedocs();                                  // zero-argument constructor

  ~Xedocs();                                 // destructor

  // ACCESSORS

  const FindStatus                           // status enum
  findXedocForClass
  (const std::string soughtClass,            // sought entity class name
   std::string&      result);                // documentation if successful, else empty

  const FindStatus                           // status enum
  findXedocForRegex
  (const std::string regex,                  // sought entity class regular expression
   std::string&      result);                // documentation if successful, else empty

  const int                                  // number of classes
  dumpClassNames
  (std::string& result);                     // newline-separated and sorted list

  // UTILITY FUNCTIONS

private:

  bool
  makeDatabase();                            // called by the constructor

  // INSTANCE DATA

private:

  const std::string                    d_fileName;         // as given in macro
  const std::string                    d_fileContents;     // as read in
```

```
  std::map<std::string, std::string>    d_database;         // key/value database

  // STATIC DATA

private:

  static logga::spLogger    s_logger;        // shared_ptr to single logger object

};

#endif // _XEDOCS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xemopt.h
//  file-create-date : Wed 20-May-2009 16:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : skeleton xem model generator / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/xemopt.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  For use with option '--xem'.

//  HEADER GUARD

#ifndef _XEMOPT_H_
#define _XEMOPT_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/xemgen.h"      // class to generate well-formatted XEM models

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <ostream>            // output streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : Xem
// ----------------------------------------------------------

class Xem
{
  // DISABLED

private:

  Xem();                                    // zero-argument constructor
  Xem(const Xem& orig);                     // copy constructor
  Xem& operator= (const Xem& orig);         // copy assignment operator

  // CREATORS

public:

  Xem
  (std::ostream& os,                        // can be either console or file ostream
   const int     svnRev,                    // usually 'xeona::svnRev'
```

```
   const int     tab = 45);                  // angle bracket alignment (usually 45 or 50)

  ~Xem();

  // MANIPULATORS

public:

  void
  head();                                   // initial material

  void
  mand                                      // mandatory entities
  (const int          steps,                // used by horizon entity
   const std::string& remark = "");         // if provided, adds trailing comment

  void
  rule
  (const std::string& annotation);

  void
  more();

  void
  tail();                                   // final material

  void
  blank();                                  // blank line

  void
  flush();                                  // flush underlying stream

  // INTERNAL DATA

private:

  std::ostream&             d_os;
  const int                 d_svnRev;
  XemGenerator              d_xemgen;

  static logga::spLogger    s_logger;       // shared_ptr to single logger objec

};

#endif // _XEMOPT_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : yeek.h
//  file-create-date : Tue 17-Nov-2009 11:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : yeek (for running extra code) value interpretation / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/yeek.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _YEEK_H_
#define _YEEK_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::yeekInterpret
// ----------------------------------------------------------
//  Description  : provide interpretation for given a 'xeona::yeek' value
//  Role         : called from function 'main' if option '--yeek' is deployed
//  Techniques   : function-like preprocessor macro, 'switch' construct
//  Status       : complete regarding code, ongoing regarding data
// ----------------------------------------------------------

namespace xeona
{
  const std::string                          // empty string if 'yeekNo' not present
  yeekInterpret
  (const unsigned yeekNo);
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::yeekSummarize
// ----------------------------------------------------------
//  Description  : provide formatted summary of all supported 'xeona::yeek' values
//  Role         : called from function 'main' if a false '--yeek' option is deployed
//  Techniques   : cycles thru 'xeona::yeekInterpret' calls
//  Status       : complete
// ----------------------------------------------------------

namespace xeona
{
```

```
  const std::string                       // contains trailing newline
  yeekSummarize
  (const int indent);

} // namespace 'xeona'

#endif // _YEEK_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : actor.h
//  file-create-date : Mon 25-Aug-2008 10:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : abstract actor entity / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/actor.h $

//  HEADER GUARD

#ifndef _ACTOR_H_
#define _ACTOR_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/costreg.h"     // cost registers

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

//  CODE

// ------------------------------------------------------------
//  CLASS           : Actor (abstract almost-base)
// ------------------------------------------------------------
//  Description  : abstract class for all actors
//  Role         : step in the inheritance web
//  Techniques   : pure virtual destructor
//  Status       : incomplete
//
//  Design notes
//
// ------------------------------------------------------------

class Actor :
  public FullEntity,
  public virtual CostRegister                 // provides the set of registers
{
  // DISABLED

private:

  Actor();                                    // zero-argument constructor
  Actor(const Actor& orig);                   // copy constructor
```

```
  Actor& operator= (const Actor& orig);      // copy assignment operator

  // CREATORS

public:

  explicit
  Actor
  (const std::string entityId,
   Record&           record);

  virtual
  ~Actor() = 0;                              // create abstract class

};

#endif // _ACTOR_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : assop.h
//  file-create-date : Mon 25-Aug-2008 12:31 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : asset operator entity / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop.h $

//  HEADER GUARD

#ifndef _ASOP_H_
#define _ASOP_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/tictoc.h"      // inherited interface for entities using common calls
#include "../b/actor.h"       // actor entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument
namespace svif { class SolverIf; }            // member function argument

class TechnicalAsset;                         // data member and function type
class LmpNode;                                // data member and function type

//  CODE

// ----------------------------------------------------------
//  ENUM            : xeona::MarketSide
// ----------------------------------------------------------
//  Description  : encode demand-side and supply-side
//  Role         : the market side is needed because demand-side bidsets require negation
//  Techniques   : (nothing special)
//  Status       : complete (if needed more generally, move to dedicated unit)
// ----------------------------------------------------------

namespace xeona
{
  enum MarketSide
    {
      e_notSpecified   = 0,
      e_demandSide     = 1,
      e_supplySide     = 2
    };
```

```
} // namespace 'xeona'

// ----------------------------------------------------------
//  CLASS           : AssetOperator (abstract)
// ----------------------------------------------------------
//  Description  : asset operator
//  Role         : operates technical assets and gateways
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class AssetOperator :
  public Actor,
  public TicToc
{
  // DISABLED

private:

  AssetOperator();                                          // zero-argument constructor
  AssetOperator(const AssetOperator& orig);                 // copy constructor
  AssetOperator& operator= (const AssetOperator& orig);     // copy assignment operator

  // CREATORS

public:

  explicit
  AssetOperator
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum);

  virtual
  ~AssetOperator() = 0;                       // create abstract class

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:

  // used to generate domain-based calls to technical assets and to LMP nodes

  std::vector<shared_ptr<TechnicalAsset> >
  getTechnicalAssets();

  virtual                                     // virtual because 'd_lmp_nodes' in not base
  std::vector<shared_ptr<LmpNode> >
  getLmpNodes();

  // standard calls

  virtual void establish();
  virtual void restructure(const xeona::DomainMode commitmentMode);
  virtual void initialize (const int step, shared_ptr<svif::SolverIf> solver);
  virtual void washup();
  virtual void conclude();

  // AUXILIARY CLASS ROUTINES

public:

  virtual
  void
  setCogenHeatWeight
  (const double cogenHeatLeadWeight);

  virtual
  const double
  getCogenHeatWeight() const;

  // INSTANCE DATA

protected:

  // tied quantities

  const std::string&                          d_technical_assets;

  // CAUTION: std::string 'd_lmp_nodes' should be set in
  // appropriate sub-classes
```

```
  // local quantities

  std::vector<shared_ptr<TechnicalAsset> >    d_technicalAssets;
  std::vector<shared_ptr<LmpNode> >           d_lmpNodes;

}; // class 'AssetOperator'

#endif // _ASOP_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asop1.h
//  file-create-date : Wed 15-Apr-2009 21:52 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete asset operators 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _ASOP1_H_
#define _ASOP1_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/auxs01.h"      // classes for auxiliary model data
#include "../b/asop.h"        // asset operator entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  FORWARD (PARTIAL) DECLARATIONS

// CAUTION: forward declarations given in the code for convenience

//  CODE

// ------------------------------------------------------------
//  CLASS           : AsopBasic
// ------------------------------------------------------------
//  Description  : basic operator
//  Role         : concrete entity
//  Techniques   : intentionally lacks an embedded optimization sub-problem (OSP)
//  Status       : complete
//
//  Design notes
//
//      A null asset operator which contributes no
//      'OspControl'-based optimization sub-problem.
//
// ------------------------------------------------------------

class AsopBasic :
  public AssetOperator
{
  // USING DECLARATIONS

protected:
```

```
  using Entity::s_logger;                   // place in common scope for this class

  // DISABLED

private:

  AsopBasic();                                      // zero-argument constructor
  AsopBasic(const AsopBasic& orig);                 // copy constructor
  AsopBasic& operator= (const AsopBasic& orig);     // copy assignment operator

  // CREATORS

public:

  explicit
  AsopBasic
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopBasic();

public:

  virtual
  const int                                 // number of technical assets processesed
  constrain
  (const xeona::DomainMode capacityMode);

}; // class 'AsopBasic'

//  ==== XEDOC ================================================
//
//  entity.asop-basic-0
//
//      class                                     > AsopBasic
//
//        basic asset operator with a null control object and
//        without internal cost formation
//
//      builtin-remark s                          <
//
//      technical-assets L                        > "teas-1 teas-2"
//
//        technical-assets in any order
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : AsopPrescribedOrder
// ------------------------------------------------------------
//  Description  : prescribed order
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : incomplete
//
//  Design notes
//
//      Prescribed merit order asset operator.
//
//      The 'technical-assets' list is given as model data in
//      DESCENDING priority.
//
// ------------------------------------------------------------

class CtlMeritOrder_A;                      // CAUTION: for class declaration typedef

class AsopPrescribedOrder :
  public AssetOperator,
  public AuxHeatLead
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:
```

```
  // CAUTION: note this simple way of swapping OSP
  // implementations, the "_X" postfix needs changing in this one
  // place only -- note also that an implementation change is
  // required to honor the same function names and signatures,
  // but not necessarily the behavior at large

  typedef CtlMeritOrder_A CtlMeritOrder;     // used for switching implementations

  // DISABLED

  AsopPrescribedOrder();                                          // zero-argument ctor
  AsopPrescribedOrder(const AsopPrescribedOrder& orig);           // copy constructor
  AsopPrescribedOrder& operator= (const AsopPrescribedOrder& orig); // copy assignment opr

  // CREATORS

public:

  explicit
  AsopPrescribedOrder
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopPrescribedOrder();

  // CALLS

public:

  virtual
  const int                                  // number of technical assets processesed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  shared_ptr<CtlMeritOrder>                  d_ctl;    // specialization required
  std::vector<shared_ptr<CtlMeritOrder> >    d_ctls;

}; // class 'AsopPrescribedOrder'

//  ==== XEDOC ================================================
//
//  entity.asop-prescribed-order-0
//
//      class                                   > AsopPrescribedOrder
//
//        asset operator implementing prescribed order operations
//
//      builtin-remark s                        <
//
//      technical-assets L                      > "teas-1 teas-2"
//
//        technical-assets given in DESCENDING priority
//
//      cogen-heat-lead-weighting [-] f         > 1.0
//
//        the cogen-lead-heat-weighting [0,1] is passed to any
//        co-generation assets to set their lead policy to heat
//        (1.0) or power (0.0) or some intermediate ratio
//
//  ===========================================================

// -----------------------------------------------------------
//  CLASS           : AsopInternalCosts
// -----------------------------------------------------------
//  Description  : least cost operator
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : incomplete / need to add 'CtlLeastCost'
//
//  Design notes
//
//      Least-cost asset operator in which the cost, one of
//      either { fin, ghg, nox, dep, luc }, is determined by the
//      domain controller.
//
```

```
//      This operator may, in the future, contributes financial
//      costs.
//
// ------------------------------------------------------

class CtlLeastCost_A;                         // CAUTION: for class declaration typedef

class AsopInternalCosts :
  public AssetOperator,
  public CostRegisterAsop
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                     // place in common scope for this class

  // TYPEDEFS

private:

  // CAUTION: note this simple way of swapping OSP
  // implementations, the "_X" postfix needs changing in this one
  // place only -- note also that an implementation change is
  // required to honor the same function names and signatures,
  // but not necessarily the behavior at large

  typedef CtlLeastCost_A CtlLeastCost;       // used for switching implementations

  // DISABLED

  AsopInternalCosts();                                          // zero-argument ctor
  AsopInternalCosts(const AsopInternalCosts& orig);             // copy constructor
  AsopInternalCosts& operator= (const AsopInternalCosts& orig);  // copy assignment opor

  // CREATORS

public:

  explicit
  AsopInternalCosts
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopInternalCosts();

public:

  virtual
  const int                                  // number of technical assets processesed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  shared_ptr<CtlLeastCost>                  d_ctl;     // specialization required
  std::vector<shared_ptr<CtlLeastCost> >    d_ctls;

}; // class 'AsopInternalCosts'

//  ==== XEDOC ================================================
//
//  entity.asop-internal-costs-0
//
//      class                                   > AsopInternalCosts
//
//        asset operator with internal cost formation
//
//      builtin-remark s                        <
//
//      technical-assets L                      > "teas-1 teas-2"
//
//        the order of the technical-assets is not significant
//
//      standing-cost-financial [$/s] f         > 5.0e-03
//      standing-cost-greenhouse [kg/s] f       > 5.0e-03
//      standing-cost-nox [kg/s] f              > 0.0
//      standing-cost-depletion [J/s] f         > 0.0
```

```
//      standing-cost-landuse [m^2/s] f          > 0.0
//
//      variable-costs-financial [$] F           < 0.0 ..
//      fixed-costs-financial [$] F              < 0.0 ..
//
//  ============================================================

#endif // _ASOP1_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asop02.h
//  file-create-date : Thu 09-Jul-2009 15:43 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete asset operators 2 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop02.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Contains asset operators needed for LMP auctions.

//  HEADER GUARD

#ifndef _ASOP02_H_
#define _ASOP02_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/auxs01.h"      // classes for auxiliary model data
#include "../b/asop.h"        // asset operator entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  CODE

// CAUTION: forward declarations given in the code for convenience

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::obtainBidsetDialog
// ------------------------------------------------------------
//  Description  : prompt user for bid information and return a bidset string
//  Role         : used by interactive bid operator 'AsopLmpBidDialog'
//  Techniques   : 'std::cin' 'boost::lexical_cast'
//  Status       : complete
//
//  Testing
//
//      This free function can be interactively tested by
//      enabling TEST 1 in the accompanying unit test.
//
// ------------------------------------------------------------

namespace xeona
{
  std::string
  obtainBidsetDialog
   (const double       capacity,             // current capacity
    const std::string& intro = "");          // optional intro text with trailing newline
```

```
}

// ----------------------------------------------------------
//  CLASS           : AsopGrid
// ----------------------------------------------------------
//  Description  : asset operator which controls LMP nodes and transmission assets
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//      Will log a warning if duty-coupling technical assets are
//      encountered.
//
// ----------------------------------------------------------

// no OSP needed

class AsopGrid :
  public AssetOperator
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // DISABLED

private:

  AsopGrid();                                     // zero-argument constructor
  AsopGrid(const AsopGrid& orig);                 // copy constructor
  AsopGrid& operator= (const AsopGrid& orig);     // copy assignment operator

  // CREATORS

public:

  explicit
  AsopGrid
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopGrid();

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:

  virtual                                   // virtual because 'd_lmp_nodes' in not base
  std::vector<shared_ptr<LmpNode> >
  getLmpNodes();

  virtual
  void
  establish();

  virtual
  const int                                 // number of technical assets processed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  // tied quantities

  const std::string&    d_lmp_nodes;

}; // class 'AsopGrid'

//  ==== XEDOC ================================================
//
//  entity.asop-grid-0
//
//      class                                   > AsopGrid
//
```

```
//        asset operator which controls LMP nodes and
//        transmission assets
//
//      builtin-remark s                        <
//
//      lmp-nodes L                             > "node-1 node-2"
//      technical-assets L                      > "teas-1 teas-2"
//
//        the technical-assets should be limited to transmission
//        assets
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : AsopLmpBidStatedTs1
// ------------------------------------------------------------
//  Description  : nodal pricing (LMP) operator using read-in bids
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : more-or-less complete
//
//  Design notes
//
//      This class utilizes a read-in bidset timeseries (the
//      "Ts") but just one (the "1").  If more than one technical
//      asset is present, the same bid is applied to both.
//
//      This class could be the starting point for another class
//      'AsopLmpBidAdaptive' which reacts adaptively based on its
//      sales history.
//
// ------------------------------------------------------------

class CtlLmpBid_A;                          // CAUTION: for class declaration typedef
class LmpBidSet;                            // data member

class AsopLmpBidStatedTs1 :
  public AssetOperator,
  public CostRegisterAsop
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:

  // CAUTION: note this simple way of swapping OSP
  // implementations, the "_X" postfix needs changing in this one
  // place only -- note also that an implementation change is
  // required to honor the same function names and signatures,
  // but not necessarily the behavior at large

  typedef CtlLmpBid_A CtlLmpBid;            // used for switching implementations

  // DISABLED

private:

  AsopLmpBidStatedTs1();                                            // zero-argument ctor
  AsopLmpBidStatedTs1(const AsopLmpBidStatedTs1& orig);             // copy constructor
  AsopLmpBidStatedTs1& operator= (const AsopLmpBidStatedTs1& orig); // copy assign opor

  // CREATORS

public:

  explicit
  AsopLmpBidStatedTs1
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopLmpBidStatedTs1();

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:
```

```
  virtual
  const int                                  // number of technical assets processed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  // tied quantities

  const std::string                          d_market_side;
  shared_ptr<std::vector<std::string> >      d_bidsets_1;

  // local quantities

  xeona::MarketSide                          d_marketSide;
  std::vector<shared_ptr<LmpBidSet> >        d_bidsets1;

  shared_ptr<CtlLmpBid>                      d_ctl;      // specialization required
  std::vector<shared_ptr<CtlLmpBid> >        d_ctls;

}; // class 'AsopLmpBidStatedTs1'

//  ==== XEDOC ================================================
//
//  entity.asop-lmp-bid-stated-ts1-0
//
//      class                                  > AsopLmpBidStatedTs1
//
//        asset operator using a single read-in (as apposed to
//        stochastic or adaptive) nodal bidding timeseries (Ts1)
//
//      builtin-remark s                       <
//
//      technical-assets L                     > "teas-1 teas-2"
//
//        the order of the technical-assets is not significant
//
//      market-side s                          > "supply"
//
//        the consumer or producer status of the operator is
//        determined by market-side in {demand,supply} respectively
//
//      bidsets-1 [W,$/J] X            > "30.0e+06 28.00e-09 * 50.0e+06 40.00e-09" ..
//
//        bidsets-1 is the sole bidset timeseries in the form of
//        (quantity band, unit price) bids in any order
//
//      standing-cost-financial [$/s] f        > 5.0e-03
//      standing-cost-greenhouse [kg/s] f      > 5.0e-03
//      standing-cost-nox [kg/s] f             > 0.0
//      standing-cost-depletion [J/s] f        > 0.0
//      standing-cost-landuse [m^2/s] f        > 0.0
//
//      variable-costs-financial [$] F         < 0.0 ..
//      fixed-costs-financial [$] F            < 0.0 ..
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : AsopLmpBidDialog
// ------------------------------------------------------------
//  Description  : nodal pricing (LMP) operator using interactive bids
//  Role         : concrete entity
//  Techniques   : interactive, 'xeona::obtainBidsetDialog'
//  Status       : more-or-less complete
//
//  Design notes
//
//      This class is interactive.
//
//      At present, no record of the submitted bidsets are kept
//      and reported -- this really should be remedied.
//
//      This class could also interact with a nominated remote
//      terminal, duly hardcoded in the data and passed to
//      function 'xeona::obtainBidsetDialog'.  See Robbins and
//      Robbins (2003 ch6 pp183-224) on UNIX special files as a
//      starting point for this exercise.
```

```
//
// ------------------------------------------------------

class CtlLmpBid_A;                          // CAUTION: for class declaration typedef
class LmpBidSet;                            // data member

class AsopLmpBidDialog :
  public AssetOperator,
  public CostRegisterAsop
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:

  // CAUTION: note this simple way of swapping OSP
  // implementations, the "_X" postfix needs changing in this one
  // place only -- note also that an implementation change is
  // required to honor the same function names and signatures,
  // but not necessarily the behavior at large

  typedef CtlLmpBid_A CtlLmpBid;            // used for switching implementations

  // DISABLED

private:

  AsopLmpBidDialog();                                          // zero-argument constructor
  AsopLmpBidDialog(const AsopLmpBidDialog& orig);              // copy constructor
  AsopLmpBidDialog& operator= (const AsopLmpBidDialog& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  AsopLmpBidDialog
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopLmpBidDialog();

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:

  virtual
  const int                                 // number of technical assets processed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  // tied quantities

  const std::string                     d_market_side;

  // local quantities

  xeona::MarketSide                     d_marketSide;
//std::vector<shared_ptr<LmpBidSet> >   d_bidsets1;

  shared_ptr<CtlLmpBid>                 d_ctl;   // specialization required
  std::vector<shared_ptr<CtlLmpBid> >   d_ctls;

}; // class 'AsopLmpBidDialog'

//  ==== XEDOC ==================================================
//
//  entity.asop-lmp-bid-dialog-0
//
//      class                                   > AsopLmpBidDialog
//
```

```
//        asset operator which interactively seeks a new bidset
//        for each interval
//
//      builtin-remark s                        <
//
//      technical-assets L                      > "teas-1 teas-2"
//
//        the order of the technical-assets is not significant
//
//      market-side s                           > "supply"
//
//        the consumer or producer status of the operator is
//        determined by market-side in {demand,supply} respectively
//
//      standing-cost-financial [$/s] f         > 5.0e-03
//      standing-cost-greenhouse [kg/s] f       > 5.0e-03
//      standing-cost-nox [kg/s] f              > 0.0
//      standing-cost-depletion [J/s] f         > 0.0
//      standing-cost-landuse [m^2/s] f         > 0.0
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
//  ============================================================

// -------------------------------------------------------------
//  CLASS           : AsopLmpBidAdaptive1
// -------------------------------------------------------------
//  Description  : adaptive nodal pricing (LMP) operator
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//      This class is adaptive.  The adaptation strategy is not
//      particularly smart and can, as currently coded, result in
//      negative prices.  It is intended as a development stub on
//      which to base more sophisticated measures.
//
// -------------------------------------------------------------

class CtlLmpBid_A;                          // CAUTION: for class declaration typedef
class LmpBidSet;                            // data member

class AsopLmpBidAdaptive1 :
  public AssetOperator,
  public CostRegisterAsop
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:

  // CAUTION: note this simple way of swapping OSP
  // implementations, the "_X" postfix needs changing in this one
  // place only -- note also that an implementation change is
  // required to honor the same function names and signatures,
  // but not necessarily the behavior at large

  typedef CtlLmpBid_A CtlLmpBid;            // used for switching implementations

  // DISABLED

private:

  AsopLmpBidAdaptive1();                                            // zero-argument ctor
  AsopLmpBidAdaptive1(const AsopLmpBidAdaptive1& orig);             // copy constructor
  AsopLmpBidAdaptive1& operator= (const AsopLmpBidAdaptive1& orig); // copy assign oper

  // CREATORS

public:

  explicit
  AsopLmpBidAdaptive1
```

```
   (const std::string entityId,
    Record&           record);

  virtual
  ~AsopLmpBidAdaptive1();

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:

  void
  establish();

  virtual
  const int                                  // number of technical assets processed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  // tied quantities

  const std::string                        d_market_side;
  const std::string                        d_openingBidset;
  const double&                            d_targetCommitment;
  const double&                            d_relativeHysteresis;
  const double&                            d_priceFactor;
  const double&                            d_priceDelta;
  shared_ptr<std::vector<std::string> >    d_submittedBidsets;

  // local quantities

  xeona::MarketSide                        d_marketSide;
  shared_ptr<LmpBidSet>                    d_bidset;   // current bidset
  std::vector<shared_ptr<LmpBidSet> >      d_bidsets;  // per technical asset

  shared_ptr<CtlLmpBid>                    d_ctl;      // specialization required
  std::vector<shared_ptr<CtlLmpBid> >      d_ctls;

}; // class 'AsopLmpBidAdaptive1'

//  ==== XEDOC ================================================
//
//  entity.asop-lmp-bid-adaptive1-0
//
//      class                                    > AsopLmpBidAdaptive1
//
//        asset operator with automated nodal bidding based on
//        the preceding and desired commitments and predetermined
//        price change rules (read the source code for details)
//
//      builtin-remark s                         <
//
//      technical-assets L                       > "teas-1"
//
//        technical-assets share the same opening bid and
//        other adaptation parameters
//
//      market-side s                            > "supply"
//
//        the consumer or producer status of the operator is
//        determined by market-side in {demand,supply} respectively
//
//      opening-bidset [W,$/J] x            > "30.0e+06 28.00e-09 * 50.0e+06 40.00e-09"
//      target-commitment [W] f             > 40.0e+06
//      relative-hysteresis [-] f           > 0.1
//      price-factor [-] f                  > 0.1
//      price-delta [$/J] f                 > 1.00e-09
//
//        opening-bidset is the starting bidset in the form of
//        (quantity band, unit price) bids in any order,
//        target-commitment is the desired duty,
//        relative-hysteresis defines the commitment zone within
//        which no adaptation occurs (zero to disable),
//        price-factor is the (non-negative) price multiplier
//        applied first (zero to disable), and price-delta is the
//        (non-negative) price increment/decrement applied second
//        (zero to disable)
//
```

```
//        for example, if a price rise in required, then:
//        bidset = bidset_preceding * (1 + price-factor) + price-delta
//
//      submitted-bidsets [W,$/J] X             < "0.0 0.0" ..
//
//        submitted-bidsets are the actual bidsets starting with
//        the opening-bidset
//
//      standing-cost-financial [$/s] f          > 5.0e-03
//      standing-cost-greenhouse [kg/s] f        > 5.0e-03
//      standing-cost-nox [kg/s] f               > 0.0
//      standing-cost-depletion [J/s] f          > 0.0
//      standing-cost-landuse [m^2/s] f          > 0.0
//
//      variable-costs-financial [$] F           < 0.0 ..
//      fixed-costs-financial [$] F              < 0.0 ..
//
//  ============================================================

#endif // _ASOP02_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asop03.h
//  file-create-date : Thu 26-Nov-2009 13:22 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete asset operators 3 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop03.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _ASOP03_H_
#define _ASOP03_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/auxs01.h"      // classes for auxiliary model data
#include "../b/asop.h"        // asset operator entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  CODE

// CAUTION: forward declarations given in the code for convenience

// ------------------------------------------------------------
//  CLASS           : AsopInelasticTs
// ------------------------------------------------------------
//  Description  : asset operator with stated demand and thereby price-inelastic behavior
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : complete but not tested
// ------------------------------------------------------------

class CtlQuan_A;                            // necessary for OSP typedef below

class AsopInelasticTs :
  public AssetOperator
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:
```

```
  typedef CtlQuan_A CtlQuan;                // used for switching implementations

  // DISABLED

private:

  AsopInelasticTs();                                        // zero-argument constructor
  AsopInelasticTs(const AsopInelasticTs& orig);             // copy constructor
  AsopInelasticTs& operator= (const AsopInelasticTs& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  AsopInelasticTs
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopInelasticTs();

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:

  virtual
  const int                                 // number of technical assets processed
  constrain
  (const xeona::DomainMode capacityMode);

  // INSTANCE DATA

private:

  // tied quantities

  const shared_ptr<std::vector<double> >    d_demands;

  // local quantities

  shared_ptr<CtlQuan>                       d_ctl;     // specialization required
  std::vector<shared_ptr<CtlQuan> >         d_ctls;

}; // class 'AsopInelasticTs'

//  ==== XEDOC ==========================================
//
//  entity.asop-inelastic-ts-0
//
//        quantifying extensity * in {J,kg,$} as appropriate
//
//      class                                   > AsopInelasticTs
//
//        asset operator with stated demand and thereby
//        price-inelastic behavior
//
//      builtin-remark s                        <
//
//      technical-assets L                      > "teas-1 teas-2"
//
//        the technical-assets should have the same quantifying
//        extenstity
//
//      demands [*/s] F                         > 5.0e+06 ..
//
//        the demand needs to be extensity-compatible with the
//        associated assets (not checked), in addition each asset
//        gets the same value
//
//  =====================================================

// -------------------------------------------------------
//  CLASS           : AsopAdaptiveTs
// -------------------------------------------------------
//  Description  : asset operator with stated demand and thereby price-inelastic behavior
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : incomplete
//
```

```
//  Design notes
//
//      This entity could be the forerunner for adaptive
//      end-users.
//
// ----------------------------------------------------------

class CtlQuan_A;                                // necessary for OSP typedef below

class AsopAdaptiveTs :
  public AssetOperator
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                       // place in common scope for this class

  // TYPEDEFS

private:

  typedef CtlQuan_A CtlQuan;                    // used for switching implementations

  // DISABLED

private:

  AsopAdaptiveTs();                                         // zero-argument constructor
  AsopAdaptiveTs(const AsopAdaptiveTs& orig);               // copy constructor
  AsopAdaptiveTs& operator= (const AsopAdaptiveTs& orig);   // copy assignment operator

  // CREATORS

public:

  explicit
  AsopAdaptiveTs
  (const std::string entityId,
   Record&           record);

  virtual
  ~AsopAdaptiveTs();

  // DOMAIN CONTROLLER POINTS OF ENTRY

public:

  virtual
  const int                                     // number of technical assets processed
  constrain
  (const xeona::DomainMode capacityMode);

  // UTILITY FUNCTIONS

private:

  bool                                          // 'true' if 'demand' was modified
  adapt1
  (shared_ptr<TechnicalAsset> ta,               // associated technical asset
   double&                    demand);          // original demand

  // INSTANCE DATA

private:

  // tied quantities

  const shared_ptr<std::vector<double> >    d_demands;
  const double&                             d_unitPriceThreshold;
  const double&                             d_adaptFactor;

  shared_ptr<std::vector<bool> >            d_curtailments;

  // local quantities

  shared_ptr<CtlQuan>                       d_ctl;     // specialization required
  std::vector<shared_ptr<CtlQuan> >         d_ctls;

}; // class 'AsopAdaptiveTs'
```

```
//  ==== XEDOC ================================================
//
//  entity.asop-adaptive-ts-0
//
//        quantifying extensity * in {J,kg,$} as appropriate
//
//      class                                   > AsopAdaptiveTs
//
//        asset operator with stated demand and with
//        price-adaptive behavior
//
//      builtin-remark s                        <
//
//      technical-assets L                      > "teas-1 teas-2"
//
//        the technical-assets should have the same quantifying
//        extenstity
//
//      demands [*/s] F                         > 5.0e+06 ..
//      unit-price-threshold [$/*] f            > 30.0e-09
//      adapt-factor [-] f                      > 0.9
//
//        the adapt-factor is envoked when the prevailing
//        marginal price trips the unit-price-threshold, this
//        causes the curtailment to become true
//
//      curtailments [-] B                      < 0 ..
//
//        the demand needs to be extensity-compatible with the
//        associated assets (not checked), in addition each asset
//        gets the same value
//
//  ============================================================

#endif // _ASOP03_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : auxs.h
//  file-create-date : Thu 16-Jul-2009 21:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : classes for auxiliary model data / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/auxs01.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Not all auxiliary model data classes are located in this unit
//  -- for example, those related to costs are elsewhere.
//
//  These classes also hold their data as private and therefore
//  need to provide suitable accessor and manipulator calls.

//  HEADER GUARD

#ifndef _AUXS01_H_
#define _AUXS01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <vector>             // STL sequence container

//  CODE

//  FORWARD (PARTIAL) DECLARATIONS

class LmpBidSet;
class Record;

// ----------------------------------------------------------
//  CLASS           : AuxBidSets
// ----------------------------------------------------------
//  Description  : support for read-in bidset timeseries
//  Role         : auxiliary data class
//  Techniques   : multiple inheritance
//  Status       : more-or-less complete
//
//  Usage
//
//      header file
//
//          #include "../b/auxs.h"
//          class SubEntity :
//            ...
//            public AuxBidSet
```

```
//          { };
//
//      implementation file
//
//          SubEntity::SubEntity
//          (const std::string entityId,
//           Record&           record) :
//            ...
//            AuxBidSet(record),
//            ...
//          { }
//
//
//  Xedoc entry
//
//      lmp-bidsets X                       > "40.0e+06 28.0e-09 * 30.0e+06 12.0e-09" ..
//
//        lmp-bidsets holds star-separated bidsets comprising
//        space-separated pairwise quantity/price [J $] offers in
//        no particular order
//
// ------------------------------------------------------------

class AuxBidSets
{
  // DISABLED

private:

  AuxBidSets();                                          // zero-argument constructor
  AuxBidSets(const AuxBidSets& orig);                    // copy constructor
  AuxBidSets& operator= (const AuxBidSets& orig);        // copy assignment operator

  // CREATORS

public:

  AuxBidSets
  (Record& record);

#ifndef _XUTEST
  virtual ~AuxBidSets() = 0;
#else
  virtual ~AuxBidSets();                      // concrete version for unit testing
#endif // _XUTEST

  // ACCESSORS (INTERPRETED RETURN)

public:

  const shared_ptr<LmpBidSet>
  getBidSet                                   // string parsed on demand
  (const int step) const;

  // INSTANCE DATA

private:

  // tied quantities

  shared_ptr<std::vector<std::string> >    d_lmpBidsets;

  // STATIC DATA

private:

  static logga::spLogger                   s_logger;   // shared_ptr to single logger obj

}; // class 'AuxBidSets'

// ------------------------------------------------------------
//  CLASS           : AuxHeatLead
// ------------------------------------------------------------
//  Description  : support for read-in heat lead weighting
//  Role         : auxiliary data class
//  Techniques   : multiple inheritance
//  Status       : more-or-less complete
//
//  Usage
//
//      Header file
```

```
//
//          #include "../b/auxs.h"
//          class SubEntity :
//            ...
//            public AuxHeatLead
//          { };
//
//      Implementation file
//
//          SubEntity::SubEntity
//          (const std::string entityId,
//           Record&           record) :
//            ...
//            AuxHeatLead(record),
//            ...
//          { }
//
//
//  Xedoc entry
//
//      cogen-heat-lead-weighting [-] f          > 1.0
//
//        the cogen-lead-heat-weighting [0,1] is passed to any
//        co-generation assets to set their lead policy to heat
//        (1.0) or power (0.0) or some intermediate ratio
//
// ---------------------------------------------------------

class AuxHeatLead
{
  // DISABLED

private:

  AuxHeatLead();                                          // zero-argument constructor
  AuxHeatLead(const AuxHeatLead& orig);                   // copy constructor
  AuxHeatLead& operator= (const AuxHeatLead& orig);       // copy assignment operator

  // CREATORS

public:

  AuxHeatLead
  (Record& record);

#ifndef _XUTEST
  virtual ~AuxHeatLead() = 0;
#else
  virtual ~AuxHeatLead();                      // concrete version for unit testing
#endif // _XUTEST

  // ACCESSORS

public:

  const double
  getCogenHeatLeadWeight() const;

  // MANIPULATORS

  void
  setCogenHeatLeadWeight
  (const double cogenHeatLeadWeighting);

  // INSTANCE DATA

private:

  // tied quantities

  double&                   d_cogenHeatLeadWeight;

  // STATIC DATA

private:

  static logga::spLogger    s_hl_logger;     // shared_ptr to single logger obj

}; // class 'AuxHeatLead'

#endif // _AUXS01_H_
```

```
//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : bandtaf.h
//  file-create-date : Tue 18-Nov-2008 09:37 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : banded tariff set and support / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/bandtaf.h $

//  HEADER GUARD

#ifndef _BANDTAF_H_
#define _BANDTAF_H_

//  AD-HOC NOTES
//
//  Design based quite closely on previously coded class
//  'LmpBidSet'.

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <utility>            // STL pair, make_pair()
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>                 // n-tuples, ref(), cref()

//  CODE

// ----------------------------------------------------------
//  CLASS           : BandedTariffSet
// ----------------------------------------------------------
//  Description  : contract-to-supply tariff set abstraction
//  Role         : used to prepare, hold, and unpack a variety of banded tariff sets
//  Techniques   : stand-alone class, 'std::pair', 'std::accumulation'
//  Status       : complete but not tested
//
//  Design notes
//
//      The actual tariffs themselves could have been implemented
//      as a standalone (nested) class, but are instead
//      represented by the private typedef 'BandedTariffSet::tariff_type'.
//
//      This class maintains two tariff set registers, one
//      poppable and the other not.  In particular:
//
//          'BandedTariffSet::clear'       empties both
//
//          'BandedTariffSet::pushString'  pushes both
```

```
//          'BandedTariffSet::pushTariff'  pushes both
//
//          'BandedTariffSet::popFirst'    pops poppable
//          'BandedTariffSet::popLast'     pops poppable
//
//          'BandedTariffSet::empty'       tests poppable size
//          'BandedTariffSet::unfilled'    tests unpoppable size and fixed charge
//
//          'BandedTariffSet::size'        returns poppable size
//          'BandedTariffSet::bands'       returns unpoppable size
//
//      If not otherwise stated, accessors offer unpoppable
//      (fixed) information.  'BandedTariffSet::size' is the only
//      exception.
//
//      In hindsight, a single labeled register may have resulted
//      in a better design.
//
// ------------------------------------------------------------

class BandedTariffSet
{
  // TYPEDEFS

private:

  typedef std::pair<double, double> tariff_type;  // namely (band, price)

  // DISABLED

private:

  BandedTariffSet(const BandedTariffSet& orig);              // copy constructor
  BandedTariffSet& operator= (const BandedTariffSet& orig);  // copy assignment operator

  // CREATORS

public:

  BandedTariffSet
  (std::string label = "");                 // optional label

  ~BandedTariffSet();

  // MANIPULATORS

  void
  clear();                                  // remove tariffs and zero the fixed charge

  std::string                               // return old label
  setLabel                                  // set new label
  (const std::string& label = "");

  int                                       // number of tariffs successfully loaded
  pushString                                // parses input, uses 'pushTariff' to load
  (const std::string sBandedTariffSet);

  int                                       // number of pushed tariffs or zero on fail
  pushTariff                                // order of insertion significant
  (const tariff_type tariff);

  void
  setSpecificFixedCharge                    // in [CUR/Ws]
  (const double specFixedCharge);

  tariff_type
  popLast();                                // pop newest tariff

  tariff_type
  popFirst();                               // pop oldest tariff

  bool                                      // 'true' indicates the 'capacity' binds
  truncate
  (const double capacity);

  // ACCESSORS

  // Clients can also access the band and price values using the
  // 'std::pair' 'first' and 'second' fields -- for example
  //
  //    double highestUnitPrice = getHighestTariff().second;
```

```
  std::string
  getLabel() const;

  double
  getSpecificFixedCharge() const;           // return fixed

  tariff_type
  getFirstOnTariff() const;

  tariff_type
  getLastOnTariff() const;

  tariff_type
  getLowestTariff() const;                  // lowest defined in terms of price

  tariff_type
  getHighestTariff() const;                 // highest defined in terms of price

  double
  getBandSum() const;                       // the contractual "capacity"

  double                                    // could be infinity
  getCapacity() const;

  bool                                      // 'true' indicates non-convex, but excluding
  isNonConvex() const;                      // .. any consideration of the fixed charge

  bool                                      // 'true' indicates not non-convex
  isConvex() const;                         // wrapper to ( isNonConvex == false )

  bool
  empty() const;                            // includes popping

  bool
  unfilled() const;                         // default fixed charge and no tariff data

  int                                       // current number of nontrivial bands
  size() const;                             // includes popping

  int                                       // current number of nontrivial bands
  bands() const;                            // excludes popping

  std::string
  summarizeAll() const;                     // for unit testing or yeek reporting

  double                                    // marginal price ($/quantity)
  getMarginalPrice
  (const double sale) const;                // transaction (quantity)

  boost::tuple                              // see 'OfrTariffSet::results_type' typedef
  <double,                                  // marginal price ($/quantity)
   double,                                  // fixed component ($)
   double,                                  // total price ($)
   double>                                  // transaction (quantity)
  interpretSale
  (const double size,
   const double sale,
   const int    interval) const;

  // INSTANCE DATA

private:

  std::string                d_label;           // optional
  double                     d_fixed;           // defaults to zero
  std::vector<tariff_type>   d_tariffset_pop;   // insertion order maintained,
  std::vector<tariff_type>   d_tariffset_fix;   // insertion order maintained
  double                     d_capacity;        // additional constraint

  // STATIC DATA

private:

  static logga::spLogger    s_logger;       // shared_ptr to single logger object

};

#endif // _BANDTAF_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : block.h
//  file-create-date : Tue 26-Aug-2008 14:21 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : abstract block entity / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/block.h $

//  HEADER GUARD

#ifndef _BLOCK_H_
#define _BLOCK_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/stats.h"       // on-the-fly statistical calculations
#include "../b/entity.h"      // entity base class plus lazy linking

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

//  CODE

// ---------------------------------------------------------
//  CLASS           : Block (abstract almost-base)
// ---------------------------------------------------------
//  Description  : abstract class for gateway and technical asset entities
//  Role         : step in the inheritance web
//  Techniques   : pure virtual destructor
//  Status       : more-or-less complete
//
//  Design notes
//
//      Definitions for duty and size
//
//          Duty covers:
//
//              - technical asset level of activity
//              - gateway sales quantity
//
//          Size covers:
//
//              - technical asset nominal capacity
//
// ---------------------------------------------------------
```

```
class Block :
  public FullEntity
{
public:

  // DISABLED

private:

  Block();                                    // zero-argument constructor
  Block(const Block& orig);                   // copy constructor
  Block& operator= (const Block& orig);       // copy assignment operator

  // CREATORS

public:

  explicit
  Block
  (const std::string entityId,
   Record&           record);

#ifndef _XUTEST
  virtual
  ~Block () = 0;                              // create abstract class
#else
  virtual
  ~Block ();                                  // allow direct usage in unit tests
#endif // _XUTEST

  // ACCESSORS

public:

  const Statistics<double>&
  getDutyStats() const;

  const Statistics<double>&
  getSizeStats() const;

  // REPORTING AND STATIC REPORTING

  const std::string
  formatStats
  (const int          indent,
   const std::string& trailingComment = "") const;

  static                                      // CAUTION: note 'static'
  const std::string
  formatStatsHeader
  (const int          indent,
   const int          ruleLength);           // note 'static' cannot be 'const'

  static                                      // CAUTION: note 'static'
  const std::string
  formatStatsFooter
  (const int          indent);               // note 'static' cannot be 'const'

  std::string
  reportDutyAndSize() const;                  // for debugging

  // INSTANCE DATA

protected:

  Statistics<double>    d_dutyStats;          // on-the-fly duty statistics
  Statistics<double>    d_sizeStats;          // on-the-fly size statistics

};

#endif // _BLOCK_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : builtins.h
//  file-create-date : Mon 28-Jan-2008 18:25 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : builtin sub-entities / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/builtins.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Support the various 'builtin' sub-entities (not to be
//  confused with the 'inbuilt' model defined in the 'c/inbuilt'
//  unit):
//
//    * TimeHorizon
//

//  HEADER GUARD

#ifndef _BUILTINS_H_
#define _BUILTINS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../b/entity.h"      // entity base class
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <iostream>           // standard io, used here for overloaded stream inserters
#include <list>               // STL sequence container
#include <string>             // C++ strings
#include <vector>             // STL sequence container

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // see "recset.h"

//  CODE

// ------------------------------------------------------------
//  CLASS           : TimeHorizon
// ------------------------------------------------------------
//  Purpose      : time horizon entity
//  Status       : complete
// ------------------------------------------------------------

class TimeHorizon :
  public Entity                               // CAUTION: inherits directly from 'Entity'
{
private:

  // DISABLED
```

```
  TimeHorizon();                                    // prevent zero-argument construction
  TimeHorizon& operator= (const TimeHorizon&);      // prevent assignment

public:

  // CREATORS

  TimeHorizon
  (const std::string entityId,              // enforced unique identifier
   Record&           record);               // associated record

  virtual                                   // support for polymorphic destruction
  ~TimeHorizon();

  void
  factoryInitialize();

  // STREAM INSERTION SUPPORT

  virtual                                   // CAUTION: needed for polymorphic behavior
  std::ostream&
  streamOut                                 // support for overloaded operator<<
  (std::ostream& os) const;

  // INTERNAL DATA

private:

  std::string&         d_builtinRemarkTh;   // as per 'FullEntity'
  Record&              d_record;            // not strictly necessary

  const int&           d_steps;
  const int&           d_interval;
  const int&           d_startHour;
  const int&           d_startDay;
  const std::string    d_hemisphere;
  xeona::Hemisphere    d_enumHemisphere;

}; // class TimeHorizon

//  ==== XEDOC ================================================
//
//  entity.time-horizon
//
//      class                                     > TimeHorizon
//
//        the TimeHorizon entity is REQUIRED and the
//        'time-horizon' identifier is MANDATORY
//
//      builtin-remark s                          <
//
//      steps [-] i                               > 6
//      interval [s] i                            > 3600
//      start-hour [-] i                          > 0
//      start-day [-] i                           > 1
//
//        the start-hour begins midnight local time and ranges
//        [0,23] and the start-day begins 01-Jan and ranges [1,365]
//
//      hemisphere s                              > "N"
//
//        the hemisphere is {N,S} for north and south
//
//        the modeler should ensuring that timeseries data given
//        elsewhere aligns with the specification given here
//
//  ============================================================

#endif // _BUILTINS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : commods.h
//  file-create-date : Wed 30-Jul-2008 07:36 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : commodities hierarchy / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/commods.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This header defines various concrete commodity classes,
//  prefixed "Cm", and the abstract 'Commodity' class from which
//  they derived.
//
//  This concept of a commodity is very general, as can be seen
//  by the definitions below.  Commodities not currently
//  supported include local labor (perhaps useful when
//  considering the system influence on local employment).

//  HEADER GUARD

#ifndef _COMMODS_H_
#define _COMMODS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/recset.h"      // record set support (also dragged in by "entity.h")
#include "../c/ghouse.h"      // global warming potential support
#include "../c/extunits.h"    // quantifying extensity enums and interpretation
#include "../b/propdata.h"    // mass-based commodity property data
#include "../b/entity.h"      // entity base class plus lazy linking

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <cmath>              // C-style maths, ceil(), floor(), sqrt()

//  FORWARD (PARTIAL) DECLARATIONS

class Gwp100Bundle;                         // function return

//  CODE

// ---------------------------------------------------------
//  CLASS           : Commodity (abstract)
// ---------------------------------------------------------
//  Description  : super-class for other commodities
//  Role         : provide commodity-common support
//  Techniques   : inheritance, abstract
```

```
//  Status       : incomplete
//
//  Design notes
//
//      Commodities can only hold intensive state information
//      (for instance, specific heating value or current
//      temperature).
//
//      The extensive state information (flows and stocks) is
//      recorded within the various 'interface' objects.
//
//      See member function 'TeasOxidToElec::establish' for the
//      dynamic cast code required to use calls to
//      sub-commodities.
//
// ----------------------------------------------------------

class Commodity :
  public FullEntity
{
  // DISABLED

private:

  Commodity();                                      // zero-argument constructor
  Commodity(const Commodity& orig);                 // copy constructor
  Commodity& operator= (const Commodity& orig);     // copy assignment operator

  // CREATORS

public:

  explicit
  Commodity
  (const std::string      entityId,
   Record&                record,
   const xeona::ExtensityUnit quantifyingExtensity);

  virtual
  ~Commodity() = 0;                             // create abstract class

  // ACCESSORS

  int                                           // minus one indicates some problem
  getInterfacePairs() const;                    // one interface pair equals one connection

  virtual
  xeona::ExtensityUnit                          // see unit 'c/extunits'
  getExtensity() const = 0;                     // abstract

  virtual
  std::string
  getExtensityStr() const = 0;                  // abstract

  // INTERNAL DATA

private:

  // note: no internal data thus far, but could add a weak census if warranted

protected:

  const xeona::ExtensityUnit    d_quantifyingExtensity;

};

// ----------------------------------------------------------
//  CLASS           : CommodityJoule (abstract)
// ----------------------------------------------------------
//  Description  : J-quantified commodity
//  Role         : step in the commodity inheritance web
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class CommodityJoule :
  public Commodity
{
  // DISABLED

private:
```

```
  CommodityJoule();                                          // zero-argument constructor
  CommodityJoule(const CommodityJoule& orig);                // copy constructor
  CommodityJoule& operator= (const CommodityJoule& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  CommodityJoule
  (const std::string entityId,
   Record&           record);

  virtual
  ~CommodityJoule() = 0;                     // create abstract class

  virtual
  xeona::ExtensityUnit
  getExtensity() const;

  virtual
  std::string
  getExtensityStr() const;

}; // class 'CommodityJoule'

// ----------------------------------------------------------
//  CLASS           : CommodityKilogram (abstract)
// ----------------------------------------------------------
//  Description  : kg-quantified commodity
//  Role         : step in the commodity inheritance web
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class CommodityKilogram :
  public Commodity
{
  // DISABLED

private:

  CommodityKilogram();                                           // zero-argument ctor
  CommodityKilogram(const CommodityKilogram& orig);              // copy constructor
  CommodityKilogram& operator= (const CommodityKilogram& orig);  // copy assignment opor

  // CREATORS

public:

  explicit
  CommodityKilogram
  (const std::string entityId,
   Record&           record);

  virtual
  ~CommodityKilogram() = 0;                  // create abstract class

  // ACCESSORS

  virtual
  xeona::ExtensityUnit
  getExtensity() const;

  virtual
  std::string
  getExtensityStr() const;

}; // class 'CommodityKilogram'

// ----------------------------------------------------------
//  CLASS           : CommodityUOA (abstract)
// ----------------------------------------------------------
//  Description  : UOA-quantified commodity
//  Role         : step in the commodity inheritance web
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class CommodityUOA :
```

```
  public Commodity
{
  // DISABLED

private:

  CommodityUOA();                                       // zero-argument constructor
  CommodityUOA(const CommodityUOA& orig);               // copy constructor
  CommodityUOA& operator= (const CommodityUOA& orig);   // copy assignment operator

  // CREATORS

public:

  explicit
  CommodityUOA
  (const std::string entityId,
   Record&           record);

  virtual
  ~CommodityUOA() = 0;                        // create abstract class

  // ACCESSORS

  virtual
  xeona::ExtensityUnit
  getExtensity() const;

  virtual
  std::string
  getExtensityStr() const;

}; // class 'CommodityUOA'

// ----------------------------------------------------------
//  CLASS           : CommodityMetreSq (abstract)
// ----------------------------------------------------------
//  Description  : m2-quantified commodity
//  Role         : step in the commodity inheritance web
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class CommodityMetreSq :
  public Commodity
{
  // DISABLED

private:

  CommodityMetreSq();                                           // zero-argument constructor
  CommodityMetreSq(const CommodityMetreSq& orig);               // copy constructor
  CommodityMetreSq& operator= (const CommodityMetreSq& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  CommodityMetreSq
  (const std::string entityId,
   Record&           record);

  virtual
  ~CommodityMetreSq() = 0;                    // create abstract class

  // ACCESSORS

  virtual
  xeona::ExtensityUnit
  getExtensity() const;

  virtual
  std::string
  getExtensityStr() const;

}; // class 'CommodityMetre2'

// ----------------------------------------------------------
//  CLASS           : CmOxidize ("oxid")
// ----------------------------------------------------------
```

```
//  Description  : oxidizable fuels commodity
//  Role         : state invariant concrete commodity
//  Quantified   : kg
//  Status       : complete
// ----------------------------------------------------------

class CmOxidize :
  public CommodityKilogram
{
  // DISABLED

private:

  CmOxidize();                                      // zero-argument constructor
  CmOxidize(const CmOxidize& orig);                 // copy constructor
  CmOxidize& operator= (const CmOxidize& orig);     // copy assignment operator

public:

  // CREATORS

  explicit
  CmOxidize
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmOxidize();

  // ACCESSORS

  virtual
  double
  getSpecCombEnthalpy() const;               // used in combination with efficiency

  virtual
  double
  getSpecCarbonDioxide() const;              // for carbon capture and later sequestration

  virtual
  double
  getSpecCo2equiv() const;                   // for release of combustion products

  virtual
  Gwp100Bundle
  getSpecGhgs() const;

  // INTERNAL DATA

protected:

  double    d_specCombustionEnthalpy;        // the modeler can elect to use LHV or HHV
  double    d_specCarbonDioxide;             // assuming capture
  double    d_specCo2equiv;                  // assuming release

};

//  ==== XEDOC ================================================
//
//  entity.cm-oxidize-0
//
//      class                                   > CmOxidize
//
//        kg-quantified state-invariant oxidizable commodity
//
//      builtin-remark s                        <
//
//      spec-combustion-enthalpy [J/kg] f       > 21.6e+06
//      spec-carbon-dioxide [kg/kg] f           > 3.7
//      spec-co2-equiv [kg/kg] f                > 3.7
//
//        the specific combustion enthalpy given here is the AR
//        (as received) HHV (higher heating value) of biocoal
//        assuming 10% moisture -- the modeler can use other
//        protocols (and substance values) but the chosen
//        protocol must be consistent with the associated
//        technical assets
//
//        spec-carbon-dioxide informs technical assets utilizing
//        carbon capture and sequestration (CCS), spec-co2-equiv
//        informs assets releasing combustion products into the
```

```
//        atmosphere
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmCarbonSeq ("cseq")
// ------------------------------------------------------------
//  Description  : carbon sequestration services carrying supercritical carbon-dioxide
//  Role         : state changeable concrete commodity
//  Quantified   : kg
//  Status       : complete
// ------------------------------------------------------------

class CmCarbonSeq :
  public CommodityKilogram
{
  // DISABLED

private:

  CmCarbonSeq();                                          // zero-argument constructor
  CmCarbonSeq(const CmCarbonSeq& orig);                   // copy constructor
  CmCarbonSeq& operator= (const CmCarbonSeq& orig);       // copy assignment operator

  // CREATORS

public:

  explicit
  CmCarbonSeq
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmCarbonSeq();

  // ACCESSORS

  virtual
  double
  getPressure
  (const int step) const;

  // MANIPULATORS

  virtual
  void
  setPressure
  (const int    step,
   const double pressure);

  // INTERNAL DATA

protected:

  shared_ptr<std::vector<double> >    d_pressures;     // transported pressure

};

//  ==== XEDOC ================================================
//
//  entity.cm-carbon-seq-0
//
//      class                                   > CmCarbonSeq
//
//        kg-quantified pressure state-changeable
//        sequestration-service commodity
//
//        demand flow is antiparallel to physical flow, similar
//        in behavior to that of a conventional fuel (like coal)
//
//      builtin-remark s                        <
//
//      pressures [Pa] F                        < 0.0 ..
//
//        the pipeline pressure is normally supercritical, say
//        around 10.0e+03
//
//  ============================================================

// ------------------------------------------------------------
```

```
//  CLASS           : CmCarbonCert ("cert")
// ---------------------------------------------------------
//  Description  : financial instrument
//  Role         : state changeable concrete commodity
//  Quantified   : kg
//  Status       : incomplete
// ---------------------------------------------------------

class CmCarbonCert :
  public CommodityKilogram
{
  // DISABLED

private:

  CmCarbonCert();                                        // zero-argument constructor
  CmCarbonCert(const CmCarbonCert& orig);                // copy constructor
  CmCarbonCert& operator= (const CmCarbonCert& orig);  // copy assignment operator

  // CREATORS

public:

  CmCarbonCert
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmCarbonCert();

  // INTERNAL DATA

protected:

  shared_ptr<std::vector<double> >    d_unitPrices;

};

//  ==== XEDOC ==================================================
//
//  entity.cm-carbon-cert-0
//
//      class                                  > CmCarbonCert
//
//        kg-quantified financial instrument
//
//      builtin-remark s                       <
//
//      unit-prices [$/kg] F                   > 14.0 ..
//
//        unit prices are currently coded as exogenous data, but
//        that could be developed
//
//  ============================================================

// ---------------------------------------------------------
//  CLASS           : CmElectricity ("elec")
// ---------------------------------------------------------
//  Description  : DC (direct current) electricity
//  Role         : concrete commodity
//  Quantified   : J (joules)
//  Techniques   : state invariant (fully specified in the model)
//  Status       : complete
// ---------------------------------------------------------

class CmElectricity :
  public CommodityJoule
{
private:

  CmElectricity();                                       // zero-argument constructor
  CmElectricity(const CmElectricity& orig);              // copy constructor
  CmElectricity& operator= (const CmElectricity& orig);  // copy assignment operator

public:

  explicit
  CmElectricity
  (const std::string entityId,
   Record&           record);
```

```
  virtual
  ~CmElectricity();

  // ACCESSORS

public:

  virtual
  double
  getVoltage() const;

  // INTERNAL DATA

protected:

  const double&    d_voltage;                // intensity

};

//  ==== XEDOC ==================================================
//
//  entity.cm-electricity-0
//
//      class                                   > CmElectricity
//
//        J-quantified electricity commodity
//
//        technical assets using electricity adopt a DC-power
//        flow model and thereby ignore shunt losses -- this
//        commodity provides suitable support
//
//      builtin-remark s                        <
//
//      voltage [V] f                           > 11.0e+03
//
//        for fixed-frequency AC power, use the RMS (root-mean-
//        square) voltage (as opposed to the peak-to-peak value)
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmWork ("work")
// ------------------------------------------------------------
//  Description  : classic "W"
//  Role         : concrete commodity
//  Quantified   : J (joules)
//  Techniques   : state invariant (fully specified in the model)
//  Status       : complete
// ------------------------------------------------------------

class CmWork :
  public CommodityJoule
{
private:

  CmWork();                                 // zero-argument constructor
  CmWork(const CmWork& orig);               // copy constructor
  CmWork& operator= (const CmWork& orig);   // copy assignment operator

public:

  explicit
  CmWork
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmWork();

  // INTERNAL DATA

protected:

};

//  ==== XEDOC ==================================================
//
//  entity.cm-work-0
//
//      class                                   > CmWork
//
```

```
//         J-quantified state-invariant classic "W" commodity,
//         covering, among other things, shaft power
//
//      builtin-remark s                          <
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmHeat ("heat")
// ------------------------------------------------------------
//  Description  : classic "Q"
//  Role         : state invariant concrete commodity
//  Quantified   : J (joules)
//  Status       : complete
// ------------------------------------------------------------

class CmHeat :
  public CommodityJoule
{
  // DISABLED

private:

  CmHeat();                                    // zero-argument constructor
  CmHeat(const CmHeat& orig);                  // copy constructor
  CmHeat& operator= (const CmHeat& orig);      // copy assignment operator

  // CREATORS

public:

  explicit
  CmHeat
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmHeat();

  // INTERNAL DATA

protected:

};

//  ==== XEDOC ================================================
//
//  entity.cm-heat-0
//
//      class                                    > CmHeat
//
//        J-quantified classic "Q" commodity
//
//        currently lacking intensive state, but could
//        specialized to include boundary temperature
//
//      builtin-remark s                          <
//
//        see also the CmThermalFluid commodity
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmThermalFluid ("thrm")
// ------------------------------------------------------------
//  Description  : recirculating thermal fluid commodity
//  Role         : state changeable concrete commodity
//  Quantified   : J (joules)
//  Status       : development stub for future use
//
//  Design notes
//
//      Provides a development stub for future use.  Will also
//      need thermal sub-network traversal support.
//
// ------------------------------------------------------------

class CmThermalFluid :
  public CommodityJoule
{
  // DISABLED
```

```
private:

  CmThermalFluid();                                          // zero-argument constructor
  CmThermalFluid(const CmThermalFluid& orig);                // copy constructor
  CmThermalFluid& operator= (const CmThermalFluid& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  CmThermalFluid
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmThermalFluid();

  // ACCESSORS

public:

  virtual
  double
  getFloTemp
  (const int step) const;

  virtual
  double
  getRetTemp
  (const int step) const;

  // MANIPULATORS

  virtual
  void
  setFloTemp
  (const int    step,
   const double temp);

  virtual
  void
  setRetTemp
  (const int    step,
   const double temp);

  virtual
  double&                                    // directly assignable
  floTemp
  (const int step);

  virtual
  double&                                    // directly assignable
  retTemp
  (const int step);

  // INTERNAL DATA

protected:

  const double&                      d_specHeatCapacity;

  shared_ptr<std::vector<double> >   d_floTemps;      // flo line, antiparallel to demand
  shared_ptr<std::vector<double> >   d_retTemps;      // ret line, parallel to demand

};

//  ==== XEDOC ==================================================
//
//  entity.cm-thermal-fluid-0
//
//      class                                  > CmThermalFluid
//
//        J-quantified temperature state-changeable recirculating
//        thermal fluid commodity
//
//      builtin-remark s                       <
//
//      spec-heat-capacity [J/kgC] f           > 4181
//
```

```
//      flo-temps [C] F                         < 0.0 ..
//      ret-temps [C] F                         < 0.0 ..
//
//        "flo" and "ret" refer to the flow and return lines of
//        the recirculating fluid, say hot water or thermal oil
//
//        provided as a development stub to enable deeco-style
//        intensive attribute setting (IAS) at some future point
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmFunds ("fund") (quantified in UOA)
// ------------------------------------------------------------
//  Description  : funds -- accounted by either the cash method or the accrual method
//  Role         : state invariant concrete commodity
//  Quantified   : $ (the dollar sign is used as a generic currency sign)
//  Status       : complete
// ------------------------------------------------------------

class CmFunds :
  public CommodityUOA
{
  // DISABLED

private:

  CmFunds();                                  // zero-argument constructor
  CmFunds(const CmFunds& orig);               // copy constructor
  CmFunds& operator= (const CmFunds& orig);  // copy assignment operator

  // CREATORS

public:

  explicit
  CmFunds
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmFunds();

  // INTERNAL DATA

protected:

};

//  ==== XEDOC ================================================
//
//  entity.cm-funds-0
//
//      class                                   > CmFunds
//
//        funds are quantified in UOA (units of account) and can
//        be accounted using either the cash method (with actual
//        transfers) or the accrual method (with legal transfers)
//
//      builtin-remark s                        <
//
//        complete but not tested
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmFission ("fiss")
// ------------------------------------------------------------
//  Description  : fissionable fuels commodity
//  Role         : state invariant concrete commodity
//  Quantified   : kg
//  Status       : incomplete - lacks intensive state support
// ------------------------------------------------------------

class CmFission :
  public CommodityKilogram
{
  // DISABLED

private:
```

```
  CmFission();                                    // zero-argument constructor
  CmFission(const CmFission& orig);               // copy constructor
  CmFission& operator= (const CmFission& orig);   // copy assignment operator

  // CREATORS

public:

  explicit
  CmFission
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmFission();

  // INTERNAL DATA

protected:

};

//  ==== XEDOC ==================================================
//
//  entity.cm-fission-0
//
//      class                                   > CmFission
//
//        kg-quantified state-invariant fissionable commodity
//
//        caution: there is currently no support for
//        nuclear-powered electricity generation
//
//      builtin-remark s                        <
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CmProductiveLand ("land")
// ------------------------------------------------------------
//  Description  : productive land commodity
//  Role         : state invariant concrete commodity
//  Quantified   : m2
//  Status       : incomplete - lacks intensive state support
// ------------------------------------------------------------

class CmProductiveLand :
  public CommodityMetreSq
{
  // DISABLED

private:

  CmProductiveLand();                                          // zero-argument constructor
  CmProductiveLand(const CmProductiveLand& orig);              // copy constructor
  CmProductiveLand& operator= (const CmProductiveLand& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  CmProductiveLand
  (const std::string entityId,
   Record&           record);

  virtual
  ~CmProductiveLand();

  // INTERNAL DATA

protected:

};

//  ==== XEDOC ==================================================
//
//  entity.cm-land-0
//
//      class                                   > CmProductiveLand
//
```

```
//        m2-quantified state-invariant land commodity
//
//        caution: not currently supported
//
//      builtin-remark s                          <
//
//  ============================================================

#endif // _COMMODS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : commods01.h
//  file-create-date : Tue 05-May-2009 18:35 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete commodities 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/commods01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _COMMODS01_H_
#define _COMMODS01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ------------------------------------------------------------
//  CLASS           : CmOxidBiocoal
// ------------------------------------------------------------
//  Description  : biocoal (formed by hydro-thermal carbonization)
//  Role         : state invariant concrete commodity
//  Quantified   : kg
//  Status       : complete
// ------------------------------------------------------------

// NOTE: this particular 'CmOxidize' specialization offers very
// little further functionality, beyond a new class name -- but
// it could well do so

class CmOxidBiocoal :
  public CmOxidize
{
  // CREATORS

public:

  explicit
  CmOxidBiocoal
  (const std::string entityId,
   Record&           record);
```

```
  virtual
  ~CmOxidBiocoal();

};

//  ==== XEDOC ========================================================
//
//  entity.cm-biocoal-0
//
//      class                                   > CmOxidBiocoal
//
//        biocoal as CmOxidize specialization with added data
//        range checking
//
//      builtin-remark s                        <
//
//      spec-combustion-enthalpy [J/kg] f       > 21.6e+06
//      spec-carbon-dioxide [kg/kg] f           > 3.6
//      spec-co2-equiv [kg/kg] f                > 3.6
//
//        the specific combustion enthalpy given here is the AR
//        (as received) HHV (higher heating value) assuming 10%
//        moisture -- the modeler can use other protocols but the
//        chosen protocol must be consistent with the associated
//        technical assets
//
//  ============================================================

#endif // _COMMODS01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : costreg.h
//  file-create-date : Mon 20-Oct-2008 08:57 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : cost registers / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/costreg.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _COSTREG_H_
#define _COSTREG_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/costs.h"       // cost sets and support

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>             // n-tuples, ref(), cref()

//  FORWARD (PARTIAL) DECLARATIONS

class CostSet;                               // see "costs.h"
class Record;                                // see "recset.h"

//  DOCUMENTATION

// ----------------------------------------------------------
//  Cost registers
// ----------------------------------------------------------
//
//  Terminology
//
//      Please note the terminology used in this unit is not
//      entirely self-evident and the material here should be
//      read IN CONJUNCTION WITH the one page cost and price
//      taxonomy table given in my (Robbie Morrison) PhD thesis
//      [and perhaps also available as file
//      'phd-costs-chart_000.odg|pdf'].
//
//  Timebase : per-second versus interval-based
//
//      For reasons of flexibility, the timebase underpinning
//      specific costs is per-second.
```

```
//
//      This approach naturally decouples specific costs data and
//      the selected horizon interval (for instance, 1800 seconds).
//
//      Cost reporting, in contrast, is typically interval-based.
//      As a rule, the horizon interval is multiplied in at the
//      last possible point in the calculation chain.  And that
//      should only occur within the 'CostRegister' class
//      hierarchy.
//
//      All other code should presume a per-second timebase in
//      relation to costs.
//
//  Class architecture
//
//      Inheritance (generalization in UML) and not composition
//      (aggregation in UML) is necessary due to the need to tie
//      back variables to the relevant record object (via the
//      'tieSingle' and 'tieTimeseries' calls).
//
//      Moreover, multiple virtual inheritance is required.  For
//      some background, see Stroustrup (1997 pp396-397).
//
//      Firstly, the base class 'CostRegister' is virtually
//      inherited by:
//
//          'TechnicalAsset'
//          'Actor'                 and thus 'DomainController'
//          'Overseer'              does not inherit from 'Actor'
//          'Gateway'
//
//      and perhaps later by the currently speculative entities:
//
//          'Zone'
//          'EconomicUnit'
//
//      The cost register base class is:
//
//          'CostRegister'          provides interface but no XEM onus
//
//      This base class architecture provides concrete entities
//      with cost registers and cost consolidation calls -- but
//      does not provide support for tying registers to XEM
//      fields and/or for undertaking specialist cost
//      calculations.
//
//      Secondly, the base class 'CostRegister' is virtually
//      inherited by the following derived cost register classes.
//      These derived cost register classes can be non-virtually
//      inherited by (normally but not necessarily) concrete
//      entities to provide the following additional
//      functionality:
//
//          'CostRegisterSRFin'     financial short-run
//          'CostRegisterSRAll'     above plus remaining short-run
//
//          'CostRegisterEmbFin'    financial embedded (via discounted cash flow)
//          'CostRegisterEmbAll'    above plus remaining embedded
//
//      Entity authors should progressively shift to more
//      encompassing cost register variants as they develop their
//      code.
//
//      Thirdly, the following specialist sub-classes give
//      suitable additional support to the top layer entities on
//      the cost consolidation tree:
//
//          'CostRegisterOverseer'  overseer support
//          'CostRegisterDomcon'    domain controller support
//          'CostRegisterAsop'      asset operator support (standing)
//
//      The first two specializations are not normally of direct
//      interest to entity authors.  The third specialization
//      will be of interest to those developing asset operators.
//
//  Tie process for record entries and local variables
//
//      With regard to data members, shared_ptr<std::vector<T> >
//      types are used as these can naturally tie with record
//      entries.
//
```

```
//  Cost consolidation process
//
//      The process of cost consolidation -- that is, the
//      assigning of technical asset and gateway costs to asset
//      operators and then to domain controllers and then to the
//      overseer singleton -- is mostly handled within the
//      function 'DomainController::consolidateDomain'.
//
//      This unit provides the means to undertake this process
//      via the various 'importCosts' and 'exportCosts' calls.
//
//  Naming convention and related information
//
//      term                                type          units
//      ------------------------------------------------------
//      financial-costs                     fin            UOA
//      revenues                            fin            UOA
//      ghgs                                ghg            kg
//      noxs                                nox            kg
//      depletable-energy-resource-usage    dep            J
//      land-usage                          luc            m^2
//      ------------------------------------------------------
//      UOA = unit of account (for instance, US dollar or European euro)
//      the ^ character indicates "to the power of"
//
//      term         kind    comment                                    issue
//      ------------------------------------------------------------------------
//      variable-    var     duty (activity) cost plus factor cost    short-run
//      fixed-       fix     nominal size (nameplate capacity) cost   short-run
//      embedded-    emb     DCF annuity cost or embedded cost        long-run
//      ------------------------------------------------------------------------
//      DCF = discounted cash flow
//
//      The notion of a factor cost is simply a duty cost which
//      arises at a gateway and not a technical asset -- simply a
//      matter of definition and nothing more.
//
//  Not all combinations useful
//
//      All naming combinations are trivially supported for
//      reasons of completeness and symmetry.  However some
//      concepts make little sense are therefore not tied to the
//      model data, for instance, 'embLuc' for embedded land
//      usage.
//
//  Adding a new cost type
//
//      Should a new cost types be required (as an illustration,
//      "lab" for labor input), then this unit, the class
//      'CostSet', and the code for switching objective functions
//      will all need modification.  Furthermore, the relevant
//      XEDOC entries will need alignment.
//
//  CAUTION: timeseries indexing
//
//      The model horizon step count is ZERO-based, as are the
//      timeseries vectors.  This convention should therefore
//      lead to straightforward code, but nonetheless keep a
//      lookout for insidious "off-by-one" bugs.
//
// ------------------------------------------------------------

//  CODE

// ------------------------------------------------------------
//  CLASS           : CostRegister (concrete)
// ------------------------------------------------------------
//  Description  : costs container base class with standard interface and processing
//  Role         : provide standardized cost formation support to entities creating costs
//  Techniques   : multiple inheritance, member initialization, record entry ties
//  Status       : first-pass complete
//
//  Design notes
//
//      This class contains the basic interface but does not add
//      any XEM onus -- meaning that no data fields are required
//      by this class definition.
//
//      Indeed, no cost vectors are bound to the model io at this
//      point, so they are simply member initialized as "reset"
//      shared pointer vectors.  That said, 'CostRegister'
```

```
//      sub-classes can then tie these variables as required.
//
//      Instance data members
//
//          A cost register currently contains the 15 following
//          data members:
//
//            { var, fix, emb } x { fin, ghg, nox, dep, luc }
//
// ------------------------------------------------------------

class CostRegister
{
  // DISABLED

private:

  CostRegister();                                       // zero-argument constructor
  CostRegister(const CostRegister& orig);               // copy constructor
  CostRegister& operator= (const CostRegister& orig);   // copy assignment operator

  // CREATORS

public:

  CostRegister
  (Record& record);

#ifdef _XUTEST
  virtual ~CostRegister();                    // unit test instantiates an object
#else
  virtual ~CostRegister() = 0;                // create abstract class
#endif // _XUTEST

  // UPDATE CALLS - virtual

protected:

  virtual                                     // redefinition not normally required
  void
  updateShortrunCosts
  (const int      step,                       // does, in theory, allow for reworking
   const CostSet& var,
   const CostSet& fix);

  virtual
  void
  updateShortrunCosts                         // wrapper to separated argument version
  (const int                               step,
   const boost::tuple<CostSet, CostSet>& shortrun);

  virtual
  void
  updateEmbeddedCosts                         // simple transfer, no calcs undertaken
  (const int      step,                       // does, in theory, allow for reworking
   const CostSet& emb);                       // calculation left to client

  virtual
  void
  updateEmbeddedCosts                         // internal calculation form
  (const int step);                           // does, in theory, allow for reworking

  // COST CONSOLIDATION CALLS

  // these cost consolidation calls are used by member functions
  // 'Overseer::run' and 'DomainController::consolidateDomain'

public:

  void
  importCosts                                 // based on "add-and-assign" operator
  (const int step,
   const boost::tuple
   <CostSet,                                  // "var" variable costs
    CostSet,                                  // "fix" fixed costs
    CostSet>& costs);                         // "emb" embedded costs, pass-by-ref required

  boost::tuple
  <CostSet,                                   // "var" variable costs
   CostSet,                                   // "fix" fixed costs
   CostSet>                                   // "emb" embedded costs
```

```
  exportCosts
  (const int step) const;

  void
  consolidateCosts                            // provided for cost consolidation
  (const int                       step,
   const shared_ptr<CostRegister> entity);   // normally a subsidiary entity

  // COST RESET CALL

  // this cost reset call is used in member functions
  // 'Overseer::run' and 'DomainController::establishDomain' and
  // also 'AssetOperator::establish'
  //
  // in hindsight, it would be better to traverse all entities
  // and make this call -- entities without meaningful registers
  // would simply return straight away

public:

  void
  resetRegister();                      // { var, fix, emb } x { fin, ghg, nox, dep, luc }

  // INSTANCE DATA

protected:

  // strictly local (non-tied) quantities

  const double    d_reset;                    // reset value set via constructor argument
  const int       d_interval;
  const int       d_horizon;

  CostSet         d_dutySpecCosts;            // for gateways, also known as unit price
  CostSet         d_sizeSpecCosts;            // normalized against nominal capacity
  CostSet         d_standingCosts;            // NOT normalized against either duty or size

  // initially local (non-tied) quantities -- which may later be
  // tied by cost register sub-classes as required -- via their
  // constructor blocks and NOT their member initialization lists

  shared_ptr<std::vector<double> >    d_varFins;  // variable costs financial
  shared_ptr<std::vector<double> >    d_fixFins;  // fixed costs financial
  shared_ptr<std::vector<double> >    d_embFins;  // capital recovery contribution
  shared_ptr<std::vector<double> >    d_varGhgs;  // variable costs greenhouse gas
  shared_ptr<std::vector<double> >    d_fixGhgs;  // fixed costs greenhouse gas
  shared_ptr<std::vector<double> >    d_embGhgs;  // embedded costs greenhouse gas
  shared_ptr<std::vector<double> >    d_varNoxs;  // variable costs NOx emissions
  shared_ptr<std::vector<double> >    d_fixNoxs;  // fixed costs NOx emissions
  shared_ptr<std::vector<double> >    d_embNoxs;  // embedded costs NOx emissions
  shared_ptr<std::vector<double> >    d_varDeps;  // variable costs depletable
  shared_ptr<std::vector<double> >    d_fixDeps;  // fixed costs depletable
  shared_ptr<std::vector<double> >    d_embDeps;  // embedded costs depletable
  shared_ptr<std::vector<double> >    d_varLucs;  // variable costs land usage
  shared_ptr<std::vector<double> >    d_fixLucs;  // fixed costs land usage
  shared_ptr<std::vector<double> >    d_embLucs;  // embedded costs land usage

//shared_ptr<std::vector<double> >    d_revenues; // revenues (like negative costs)

// STATIC DATA

protected:

  static logga::spLogger s_logger_costreg;   // shared_ptr to single logger object

}; // class 'CostRegister'

// ----------------------------------------------------------
//  CLASS           : CostRegisterOverseer
// ----------------------------------------------------------
//  Description  : builds on 'CostRegister' to add support for the overseer
//  Role         : use by the class 'Overseer' singleton
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  XEM fields
//
//      variable-costs-financial [$] F           < 0.0 ..
//      fixed-costs-financial [$] F              < 0.0 ..
//      embedded-costs-financial [$] F           < 0.0 ..
//      variable-costs-greenhouse [kg] F         < 0.0 ..
```

```
//      fixed-costs-greenhouse [kg] F           < 0.0 ..
//      embedded-costs-greenhouse [kg] F        < 0.0 ..
//      variable-costs-nox [kg] F               < 0.0 ..
//      fixed-costs-nox [kg] F                  < 0.0 ..
//      embedded-costs-nox [kg] F               < 0.0 ..
//      variable-costs-depletion [J] F          < 0.0 ..
//      fixed-costs-depletion [J] F             < 0.0 ..
//      embedded-costs-depletion [J] F          < 0.0 ..
//      variable-costs-landuse [m^2] F          < 0.0 ..
//      fixed-costs-landuse [m^2] F             < 0.0 ..
//      embedded-costs-landuse [m^2] F          < 0.0 ..
//
//      total-financial [$] f                   < 0.0
//      total-greenhouse [kg] f                 < 0.0
//      total-nox [kg] f                        < 0.0
//      total-depletion [J] f                   < 0.0
//      total-landuse [m^2] f                   < 0.0
//
// ------------------------------------------------------------

class CostRegisterOverseer :
  public virtual CostRegister               // 'virtual' prevents double registers
{
  // DISABLED

private:

  CostRegisterOverseer();                                                 // zero-arg ctor
  CostRegisterOverseer(const CostRegisterOverseer& orig);                 // copy constructor
  CostRegisterOverseer& operator= (const CostRegisterOverseer& orig);  // copy ass opor

  // CREATORS

public:

  CostRegisterOverseer
  (Record& record);

  virtual ~CostRegisterOverseer() = 0;       // create abstract class

  // COST CONSOLIDATION CALLS

public:

  void
  importCosts                               // signature different from base version
  (const int step,
   const CostSet&  var,
   const CostSet&  fix,
   const CostSet&  emb);

  // INSTANCE DATA

protected:

  double&    d_totalFin;
  double&    d_totalGhg;
  double&    d_totalNox;
  double&    d_totalDep;
  double&    d_totalLuc;

}; // class 'CostRegisterOverseer'

// ------------------------------------------------------------
//  CLASS           : CostRegisterDomcon
// ------------------------------------------------------------
//  Description  : builds on 'CostRegister' to add support for domain controllers
//  Role         : use by class 'DomainController'
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  XEM fields
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
// ------------------------------------------------------------

class CostRegisterDomcon :
  public virtual CostRegister               // 'virtual' prevents double registers
{
```

```
  // DISABLED

private:

  CostRegisterDomcon();                                          // zero-arg ctor
  CostRegisterDomcon(const CostRegisterDomcon& orig);            // copy constructor
  CostRegisterDomcon& operator= (const CostRegisterDomcon& orig);  // copy ass opor

  // CREATORS

public:

  CostRegisterDomcon
  (Record& record);

  virtual ~CostRegisterDomcon() = 0;         // create abstract class

}; // class 'CostRegisterDomcon'

// ----------------------------------------------------------
//  CLASS           : CostRegisterAsop
// ----------------------------------------------------------
//  Description  : builds on 'CostRegister' to add standing financial support
//  Role         : drop-in replacement
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  Overview - short-run financial support
//
//      This class provides additional support for short-run
//      (variable and fixed) financial costs.
//
//  XEM fields
//
//      standing-cost-financial [$/s] f          > 200.0
//      standing-cost-greenhouse [kg/s] f        > 200.0
//      standing-cost-nox [kg/s] f               > 200.0
//      standing-cost-depletion [J/s] f          > 200.0
//      standing-cost-landuse [m^2/s] f          > 200.0
//
//      variable-costs-financial [$] F           < 0.0 ..
//      fixed-costs-financial [$] F              < 0.0 ..
//
// ----------------------------------------------------------

class CostRegisterAsop :
  public virtual CostRegister                // 'virtual' prevents double registers
{
  // DISABLED

private:

  CostRegisterAsop();                                          // zero-argument ctor
  CostRegisterAsop(const CostRegisterAsop& orig);              // copy constructor
  CostRegisterAsop& operator= (const CostRegisterAsop& orig);  // copy assignment operator

  // CREATORS

public:

  CostRegisterAsop
  (Record& record);

  virtual ~CostRegisterAsop() = 0;          // create abstract class

  // INSTANCE DATA

 protected:

  // tied quantities

  const double&    d_standingFin;            // standing cost financial
  const double&    d_standingGhg;            // standing cost greenhouse
  const double&    d_standingNox;            // standing cost nox
  const double&    d_standingDep;            // standing cost depletion
  const double&    d_standingLuc;            // standing cost land use

}; // class 'CostRegistedAsop'

// ----------------------------------------------------------
//  CLASS           : CostRegisterSRFin
```

```
// ----------------------------------------------------------
//  Description  : builds on 'CostRegister' to add short-run financial support
//  Role         : drop-in replacement
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  Overview - short-run financial support
//
//      This class provides additional support for short-run
//      (variable and fixed) financial costs.
//
//  XEM fields (where . = duty units x s, * = capacity units)
//
//      nameplate-capacity [*] f                > 200
//      duty-specific-cost-financial [$/.] f    > 2.0
//      size-specific-cost-financial [$/*/s] f  > 3.0
//      standing-cost-financial [$/s] f         > 5.0
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
//  CAUTION: upload functions
//
//      The various 'CostRegister::uploadShortrunCost' functions
//      need to be repeated here.
//
// ----------------------------------------------------------

class CostRegisterSRFin :
  public virtual CostRegister               // 'virtual' prevents double registers
{
  // DISABLED

private:

  CostRegisterSRFin();                                          // zero-argument ctor
  CostRegisterSRFin(const CostRegisterSRFin& orig);             // copy constructor
  CostRegisterSRFin& operator= (const CostRegisterSRFin& orig);  // copy assignment opor

  // CREATORS

public:

  CostRegisterSRFin
  (Record& record);

  virtual ~CostRegisterSRFin() = 0;          // create abstract class

  // UPDATE CALLS - for the given horizon step

protected:

  virtual
  void
  updateShortrunCosts                       // wrapper to separated argument version
  (const int                              step,
   const boost::tuple<CostSet, CostSet>& shortrun);

  virtual
  void
  updateShortrunCosts                       // excludes embedded costs financial
  (const int    step,                       // does, in theory, allow for reworking
   const double currentDuty);               // duty includes gateway transactions

  // INSTANCE DATA

 protected:

  // tied quantities

  const double&    d_nameplateSize;         // for size cost calculations
  const double&    d_dutySpecFin;           // duty-specific cost financial
  const double&    d_sizeSpecFin;           // size-specific cost financial
  const double&    d_standingFin;           // standing cost financial

}; // class 'CostRegistedSRFin'

// ----------------------------------------------------------
//  CLASS          : CostRegisterSRAll
// ----------------------------------------------------------
//  Description  : builds on 'CostRegister' to add short-run non-financial support
```

```
//  Role         : drop-in replacement
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  Overview - short-run financial support
//
//      This class provides additional support for short-run
//      non-financial costs.
//
//  XEM fields (where . = duty units x s, * = capacity units)
//
//      duty-specific-cost-greenhouse [kg/.] f   > 1.0
//      size-specific-cost-greenhouse [kg/*/s] f > 2.0
//      standing-cost-greenhouse [kg/s] f        > 3.0
//      duty-specific-cost-nox [kg/.] f          > 1.0
//      size-specific-cost-nox [kg/*/s] f        > 2.0
//      standing-cost-nox [kg/s] f               > 3.0
//      duty-specific-cost-depletion [J/.] f     > 1.0
//      size-specific-cost-depletion [J/*/s] f   > 2.0
//      standing-cost-depletion [J/s] f          > 3.0
//      duty-specific-cost-landuse [m^2/.] f     > 1.0
//      size-specific-cost-landuse [m^2/*/s] f   > 2.0
//      standing-cost-landuse [m^2/s] f          > 3.0
//
//      variable-costs-greenhouse [kg] F         < 0.0 ..
//      fixed-costs-greenhouse [kg] F            < 0.0 ..
//      variable-costs-nox [kg] F                < 0.0 ..
//      fixed-costs-nox [kg] F                   < 0.0 ..
//      variable-costs-depletion [J] F           < 0.0 ..
//      fixed-costs-depletion [J] F              < 0.0 ..
//      variable-costs-landuse [m^2] F           < 0.0 ..
//      fixed-costs-landuse [m^2] F              < 0.0 ..
//
// -----------------------------------------------------

class CostRegisterSRAll :
  public CostRegisterSRFin
{
  // DISABLED

private:

  CostRegisterSRAll();                                          // zero-argument ctor
  CostRegisterSRAll(const CostRegisterSRAll& orig);             // copy constructor
  CostRegisterSRAll& operator= (const CostRegisterSRAll& orig);  // copy assignment opor

  // CREATORS

public:

  CostRegisterSRAll
  (Record& record);

  virtual ~CostRegisterSRAll() = 0;          // create abstract class

  // UPDATE CALLS - for the given horizon step

protected:

  virtual
  void
  updateShortrunCosts                        // excludes embedded costs financial
  (const int    step,                        // does, in theory, allow for reworking
   const double currentDuty);

  // INSTANCE DATA

 protected:

  // tied quantities

  const double&    d_dutySpecGhg;            // duty-specific cost greenhouse
  const double&    d_sizeSpecGhg;            // size-specific cost greenhouse
  const double&    d_standingGhg;            // standing cost greenhouse
  const double&    d_dutySpecNox;            // duty-specific cost nox
  const double&    d_sizeSpecNox;            // size-specific cost nox
  const double&    d_standingNox;            // standing cost nox
  const double&    d_dutySpecDep;            // duty-specific cost depletion
  const double&    d_sizeSpecDep;            // size-specific cost depletion
  const double&    d_standingDep;            // standing cost depletion
  const double&    d_dutySpecLuc;            // duty-specific cost land usage
```

```
  const double&    d_sizeSpecLuc;            // size-specific cost land usage
  const double&    d_standingLuc;            // standing cost land usage

}; // class 'CostRegistedSRAll'

// ---------------------------------------------------------
//  CLASS           : CostRegisterEmbFin
// ---------------------------------------------------------
//  Description  : adds DCF support
//  Role         : drop-in replacement
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  Note
//
//      DCF is discounted cash flow (analysis)
//
//  Overview - additional DCF support
//
//      This class provides additional support for discounted
//      cash flow analysis.
//
//  XEM fields
//
//      annual-discount-rate-decimal [-] f       > 0.10
//      economic-life [y] i                      > 20
//      capex-initial [$] f                      > 120e+03
//      capex-terminal [$] f                     >  40e+03
//      current-age [y] i                        > 2
//
//        a negative 'capex-terminal' indicates salvage revenue
//        as opposed to decommissioning cost
//
//      embedded-costs-financial [$] F           < 0.0 ..
//
//        the 'embedded-costs-financial' are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//  CAUTION: upload functions
//
//      The various 'CostRegister::uploadEmbeddedCost' functions
//      need to be repeated here.
//
// ---------------------------------------------------------

class CostRegisterEmbFin :
  public virtual CostRegister
{
  // DISABLED

private:

  CostRegisterEmbFin();                                          // zero-argument ctor
  CostRegisterEmbFin(const CostRegisterEmbFin& orig);            // copy constructor
  CostRegisterEmbFin& operator= (const CostRegisterEmbFin& orig);  // copy assignment opor

  // CREATORS

public:

  CostRegisterEmbFin
  (Record& record);

  virtual ~CostRegisterEmbFin() = 0;         // create abstract class

  // UPDATE CALLS - for the given horizon step

  virtual
  void
  updateEmbeddedCosts                        // includes DCF financial cost calculations
  (const int step);                          // does, in theory, allow for reworking

  // INSTANCE DATA

protected:

  // tied quantities

  const double&    d_annualDiscountRate;     // risk-adjusted interest per annum (decimal)
  const int&       d_economicLife;           // economic life in years (not seconds)
```

```
  const double&    d_capitalInputInitial;    // capital input at start of economic life
  const double&    d_capitalInputTerminal;   // capital input at end of economic life
  const int&       d_currentAge;             // current age

}; // class 'CostRegisterEmbFin'

// -----------------------------------------------------------
//  CLASS           : CostRegisterEmbAll
// -----------------------------------------------------------
//  Description  : adds DCF support
//  Role         : drop-in replacement
//  Techniques   : see overarching documentation
//  Status       : first-pass complete
//
//  Note
//
//      DCF is discounted cash flow (analysis)
//
//  Overview - additional DCF support
//
//      This class provides additional support for discounted
//      cash flow analysis.
//
//  XEM fields
//
//      annual-discount-rate-decimal [-] f       > 0.10
//      economic-life [y] i                      > 20
//      capex-initial [$] f                      > 120e+03
//      capex-terminal [$] f                     >  40e+03
//      current-age [y] i                        > 2
//
//        a negative 'capex-terminal' indicates salvage revenue
//        as opposed to decommissioning cost
//
//      embedded-costs-financial [$] F           < 0.0 ..
//
//        the 'embedded-costs-financial' are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//      physical-life [y] i                      > 30
//      investment-greenhouse [kg] f             > 3.0e+03
//      investment-nox [kg] f                    > 0.0
//      investment-depletion [J] f               > 3.0e+03
//      investment-landuse [m^2] f               > 0.0
//
//      embedded-costs-greenhouse [kg] F         < 0.0 ..
//      embedded-costs-nox [kg] F                < 0.0 ..
//      embedded-costs-depletion [J] F           < 0.0 ..
//      embedded-costs-landuse [m^2] F           < 0.0 ..
//
// -----------------------------------------------------------

class CostRegisterEmbAll :
  public CostRegisterEmbFin
{
  // DISABLED

private:

  CostRegisterEmbAll();                                          // zero-argument ctor
  CostRegisterEmbAll(const CostRegisterEmbAll& orig);            // copy constructor
  CostRegisterEmbAll& operator= (const CostRegisterEmbAll& orig);  // copy assignment opor

  // CREATORS

public:

  CostRegisterEmbAll
  (Record& record);

  virtual ~CostRegisterEmbAll() = 0;         // create abstract class

  // UPDATE CALLS - for the given horizon step

  virtual
  void
  updateEmbeddedCosts                        // includes DCF financial cost calculations
  (const int step);                          // does, in theory, allow for reworking

  // INSTANCE DATA
```

```
protected:

  // tied quantities

  const int&       d_physicalLife;           // physical life in years (not seconds)
  const double&    d_investGhg;              // greenhouse investment
  const double&    d_investNox;              // nox investment
  const double&    d_investDep;              // depletion investment
  const double&    d_investLuc;              // land use investment

}; // class 'CostRegisterEmbAll'

#endif // _COSTREG_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : domcon.h
//  file-create-date : Thu 07-Aug-2008 20:51 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain controller entity / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/domcon.h $

//  HEADER GUARD

#ifndef _DOMCON_H_
#define _DOMCON_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../b/costreg.h"     // cost registers
#include "../b/actor.h"       // actor entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <map>                // STL associative container
#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>             // n-tuples, ref(), cref()

//  FORWARD (PARTIAL) DECLARATIONS

class AssetOperator;                         // data member
class DemandJunction;                        // data member
class Block;                                 // member function return
class Gateway;                               // member function return

class Record;                                // constructor argument
namespace svif { class SolverIf; }           // data member
class CostSet;                               // member function argument

//  CODE

// ----------------------------------------------------------
//  CLASS          : DomainController
// ----------------------------------------------------------
//  Description  : controlling entity for commitment domains
//  Role         : sits under overseer, contains 'capset' and 'transolve' calls
//  Techniques   : each 'DomainController' object contains its own MIP problem interface
//  Leading tag  : domcon
//  Status       : incomplete
//
//  Design notes
```

```
//
//      Commitment mode (aka control strategy)
//
//          The relevant 'domcon' sets the control strategy for
//          its domain via the commitment mode.  The commitment
//          mode itself is encoded in the enum
//          'xeona::DomainMode', as defined in 'f/ospmode.h'
//
//          Commitment modes are set as part of the
//          'DomainController' definition and, in some cases,
//          using read-in model data.
//
//          The commitment mode value is currently invariant, but
//          the design does allow for the commitment mode to be
//          reset partway thru a simulation.
//
//          Users of commitment modes (most notably OSPs) can
//          also encode an acceptable set of control strategies,
//          for instance:
//
//              int d_commitmentModeSum
//                = xeona::e_shortrunFin
//                | xeona::e_shortrunGhg;
//
//          Each recipient can check for commitment mode
//          miscibility (internal compatibility) as follows:
//
//              const tribool ret
//                = xeona::isTwoContained(offeredCommitmentMode,
//                                        d_commitmentModeSum)
//
//          Immiscibility errors should be logged as warnings.
//          Assuming properly coded modules, an immiscibility
//          error indicates a fault with the model (which the
//          modeler should fix).
//
//      "lowest" for member functions
//
//          For the member functions here, "lowest" means the
//          least-ranked (zero-based) index for the given gate:
//
//            selgates: in descending priority
//            buygates: in arbitrary but consistent order
//
//          Recall that selgates are given by the modeler,
//          whereas buygates are revealed by the
//          'xeona::registerGates' call.
//
// ------------------------------------------------------------

class DomainController :
  public Actor,
  public CostRegisterDomcon
{
  // DISABLED

private:

  DomainController();                                          // zero-argument constructor
  DomainController(const DomainController& orig);              // copy constructor
  DomainController& operator= (const DomainController& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  DomainController
  (const std::string entityId,
   Record&           record);

  virtual
  ~DomainController();

  // ACCESSORS

public:

  int                                        // number of selgates (from input data)
  selgateCount() const;
```

```
int                                     // number of buygates (needs DFS traversal)
buygateCount() const;

// OVERSEER POINTS OF ENTRY

void
establishDomain();                      // beginning of horizon call

void
restructureDomain();                    // structure change call

void
initializeDomain                        // beginning of interval call
(const int step);

void
washupDomain();                         // end of interval call

void
consolidateDomain                       // dedicated consolidate costs call
(const int step,
 CostSet&  var,
 CostSet&  fix,
 CostSet&  emb);

void
concludeDomain();                       // end of horizon call

// REGISTER GATEWAYS FUNCTION POINTS OF ENTRY

// used by helper function '::registerGateways' called in turn
// by free function 'xeona::registerGates' called in turn by
// 'xeona::simulate'

std::map<shared_ptr<Gateway>, shared_ptr<DomainController> >
mapGatesToDomain();

std::map<shared_ptr<Gateway>, shared_ptr<Block> >
mapGatesToBlocks();

std::map<shared_ptr<Block>, shared_ptr<DomainController> >
mapBlocksToDomain();

int                                     // number of buygates following insertion
addBuygate
(shared_ptr<Gateway> buygate);

std::vector<shared_ptr<Gateway> >
retBuygates();

std::vector<shared_ptr<Gateway> >
retSelgates();

// DOMAIN FUNCTIONS

// used by class 'TravDepthFirst'

bool getHyphen() const;
bool markHyphen();
void resetHyphen();

// GET GATEWAY FUNCTIONS

// used by class 'TravDepthFirst'

shared_ptr<Gateway> lowestNoTildeSel();
void                pushTilde(shared_ptr<Gateway> gate);
shared_ptr<Gateway> popTilde();

// used by functors derived from class 'CapTransAlg'

// CAUTION: the commented out functions fit the functionality
// pattern but are not required -- however they should remain
// in the code because future CTA developments may need them

shared_ptr<Gateway> lowestNoThetaBuy();
shared_ptr<Gateway> lowestNoStarBuy();

shared_ptr<Gateway> lowestNotTransBuy();
shared_ptr<Gateway> lowestNotTransSel();
```

```
//shared_ptr<Gateway> lowestNotCapBuy();
//shared_ptr<Gateway> lowestNotCapSel();

  shared_ptr<Gateway> lowestUnderCapBuy();
  shared_ptr<Gateway> lowestUnderCapSel();

//shared_ptr<Gateway> lowestBuy();
  shared_ptr<Gateway> lowestSel();

  // RESET GATEWAY FUNCTIONS

  void resetTildeSelgates();

  // CAPSET AND TRANSOLVE FUNCTIONS

  // 'capset' uses minimum and maximum network flow problem
  // formulations to determine gateway capacities dynamically
  // (static capset is yet to be designed and implemented)

  bool                                      // return 'false' on failure
  capset
  (const int               step,
   shared_ptr<Gateway>     targetGateway,   // target gateway
   const xeona::DomainMode capacityMode);   // from overseer

  // 'transolve' uses a least cost network flow formulation to
  // determine a minimum shortrun cost solution

  bool                                      // return 'false' on failure
  transolve
  (const int               step,
   const xeona::DomainMode commitmentMode);

  // INTERNAL DATA

private:

  // tied quantities : read-in GLPK settings

  const bool&                               d_initEmployScaling;
  const bool&                               d_initEmployAdvIniBasis;
  const bool&                               d_initSimplexPresolver;
  const bool&                               d_initIntegerPresolver;

  // tied quantities and local quantities

  std::string&                              d_commitment_mode;
  xeona::DomainMode                         d_commitmentMode;

  std::string&                              d_ranked_selgates;
  std::vector<shared_ptr<Gateway> >         d_rankedSelgates;

  std::string&                              d_asset_operators;
  std::vector<shared_ptr<AssetOperator> >   d_assetOperators;

  std::string&                              d_demand_junctions;
  std::vector<shared_ptr<DemandJunction> >  d_demandJunctions;

  // local quantities

  int                                       d_step;
  shared_ptr<svif::SolverIf>                d_solver;

  std::vector<shared_ptr<Gateway> >         d_randomBuygates;

  std::vector<shared_ptr<Gateway> >         d_tildePushPop;
  bool                                      d_hyphen;      // DFS revisit lock

}; // class 'DomainController'

//  ==== XEDOC ================================================
//
//  entity.domain-controller-0
//
//      class                                    > DomainController
//
//        a domain controller entity (the only one provided)
//        which can take one of a number of commitment modes --
//        but REQUIRES that the managed entities support the
//        elected mode
//
```

```
//      builtin-remark s                        <
//
//      init-scale-problem b                    > 1
//      init-use-advanced-initial-basis b       > 1
//      init-use-simplex-presolver b            > 1
//      init-use-mip-presolver b                > 0
//
//        these four GLPK solver behavior settings should be set
//        to true unless run-time tests indicate otherwise
//
//      commitment-mode s                       > "fin"
//
//        supported commitment-mode values (lmp is nodal pricing):
//        fin | ghg | nox | dep | luc | lmp | merit | first
//
//      ranked-selgates L                       > "gate-2 gate-1"
//
//        the ranked-selgates (bridging the right side) must be
//        null, one, or listed in DESCENDING priority
//
//      asset-operators L                       > "asop-1 asop-2"
//
//        the asset-operators may be null, individual, or listed
//        in no particular order
//
//      demand-junctions L                      > "teas-demand-2-split-0"
//
//        the demand-junctions, which split and join demand, may
//        be null, individual, or listed in no particular order
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
//  ============================================================

#endif // _DOMCON_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : entity.h
//  file-create-date : Fri 15-Jun-2007 08:51 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : entity base class plus lazy linking / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/entity.h $

//  HEADER GUARD

#ifndef _ENTITY_H_
#define _ENTITY_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/xeona_ptr.h"   // remappable counted pointer which mimics shared_ptr
#include "../c/linklog.h"     // utility class to record entity linking results
#include "../a/logger.h"      // run-time logging functionality (as required)

#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between TR1 and Boost smart pointers

#include <iostream>           // standard io, used here for overloaded stream inserters
#include <list>               // STL sequence container
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <typeinfo>           // run-time type info, NOTE: passive reporting role only

//  NAMESPACE DIRECTIVES

using xeona::assign_ptr;                    // remappable counted pointer

//  FORWARD (PARTIAL) DECLARATIONS

class Overseer;                             // see "overseer.h"
class DomainController;                     // see "domcon.h"
class Record;                               // see "recset.h"

//  CODE

// ---------------------------------------------------------
//  ENUM            : xeona::Hemisphere
// ---------------------------------------------------------

namespace xeona
{
  enum Hemisphere
    {
      e_notSet = 0,
```

```
      e_north  = 1,
      e_south  = 2
    };
}

// ----------------------------------------------------------
//  CLASS           : Entity (abstract base)
// ----------------------------------------------------------
//  Description  : abstract base class for all entities
//  Role         : provide basic housekeeping and class-wide support
//  Status       : working and mostly complete
// ----------------------------------------------------------
//
//  Design notes
//
//      Overview
//
//          There are two programmatic issues that the 'Entity'
//          class needs to deal with at the base-class level:
//
//            * the factory construction of entities from model data
//            * the linking in of associated entities
//
//      Factory construction
//
//          Refer to documentation for the 'EntityFactory' class
//          and for the '::createType<>' free function template.
//
//      Lazy linking
//
//          The term "link" here refers an internal procedure for
//          remapping hollow entities to full entities and NOT to
//          the compile-time process of combining object files to
//          create an executable file.
//
//          Client sub-entities often need to link to service
//          sub-entities by way of embedded applied pointers.
//          These linkages cannot be established when the clients
//          are constructed because the respective service
//          sub-entities may not have been built yet.  The
//          process of linking, therefore, must wait until the
//          process of sub-entity construction is complete.
//
//          Applied pointers are remappable shared pointers.
//          Applied pointers are unique to 'xeona'.  See unit
//          'xeona_ptr' for details.
//
//          This code employs "lazy linking".  Lazy linking
//          allows an intuitive and economical syntax to be used
//          by entity authors when specifying these linkages --
//          because the actual mechanics of linking is handled
//          outside of the modules.
//
//          The service entity must be a "LLE" or "lazy-linkable"
//          entity.  There are two LLE requirements:
//
//              * a locally defined 'polylink' member function
//                which comprises one line of standard code with
//                the appropriate template argument
//
//              * a single-argument constructor which lacks the
//                'Record&' argument and which is supported up
//                (base-wise) the inheritance chain
//
//          The module syntax is as follows (see the ambient air
//          statement) where 'd_ambientAirContext' is of type
//          'assign_ptr<CxAmbientAir>':
//
//              SubEntity::SubEntity
//              (const std::string& entityId,
//               Record&            record) :
//                Entity(entityId, record),
//                d_coeff(record.tieSingle<double>("coeff")),
//                d_ambientAirContext(record.lazyLink<CxAmbientAir>
//                  ("ambient-air-context"))
//              {
//              }
//
//          The "link" entity CONSTRUCT chain is as follows:
//
//            client sub-entity constructor (see above)
```

```
//             |
//             + assign_ptr<Entity> Record::lazyLink<E>(linkname)  // name as record field
//                |
//                + static shared_ptr<Entity> makeLinkEntity(soughtId)  // recovered id
//                   |
//                   - assign_ptr<E> link(new E(soughtId))
//                   - s_censusLink.push_back(link)
//                   - return link
//
//          And the "link" entity LINKING chain is as follows:
//
//            simulation management
//             |
//             + static Entity::linkAll()
//
//      Function template 'polylink' in standard-form:
//
//          private:
//            bool
//            LLE::polylink(assign_ptr<Entity>& pass)
//            {
//              return pass.revamp(retFull<LLE>());    // key call
//            }
//
// ------------------------------------------------------------

class Entity
  : public enable_shared_from_this<Entity>
{
  // FRIENDS

  friend class TimeHorizon;                 // sub-entity sets horizon steps and interval
  friend class Interface;                   // for access to 'retSharedPtr'
  friend class BuySide;                     // for access to 'retSharedPtr', gateway
  friend class SelSide;                     // for access to 'retSharedPtr', gateway
  friend class LinkLogger;                  // for access to 'retSharedPtr'

  friend class FullEntity;                  // to retain private access specifiers

  // DISABLED

private:

  Entity();                                 // zero-argument construction
  Entity(const Entity& orig);               // copy constructor
  Entity& operator= (const Entity& orig);   // copy assignment operator

  // CREATORS

public:

  explicit
  Entity                                    // used by "link" factory ('makeLinkEntity')
  (const std::string entityId) :
    d_identifier(entityId),
    d_record(s_recordStandIn),
    d_typeStr()
  { }

  explicit                                  // disable implicit type conversions (string)
  Entity                                    // used by "full" factory (complicated)
  (const std::string entityId,              // enforced unique identifier on full list
   Record&           record);               // associated record

protected:

  // PROTECTED VIRTUAL DESTRUCTORS AND SMART POINTERS:
  //
  // First, the use of 'virtual' destructors means that the
  // correct destructors are called.
  //
  // Second, the use of 'protected' rather than 'public' means
  // that a raw pointer (extracted from a smart pointer, say)
  // cannot use 'delete', but (the original) smart pointer still
  // works fine -- see Karlsson (2006 pp23-24) for an explanation
  // and also Sutter and Alexandrescu (2005 pp90-91).
  //
  // Concrete classes need 'public' destructors and should also
  // be 'virtual' if further derived.

  virtual                                   // support for polymorphic destruction
```

```
  ~Entity() = 0;                            // create abstract class

  // LINKING CALL NON-OVERWRITE WARNING

protected:

  virtual
  bool
  polylink(assign_ptr<Entity>& pass)
  {
    // key call is "return pass.revamp(retFull<L>())" with L = lazy-linkable entity

    // logging messages only
    s_logger->repx(logga::warn, "function not overwritten as required", "Entity");
    std::ostringstream put;
    put << "  lazy-linkable sub-entities must overwrite the 'polylink' function" << "\n";
    s_logger->putx(logga::dbug, put);

    // return fail
    return false;
  }

  // INITIALIZERS

public:

  // INITIALIZATION: the object factory invokes the the
  // 'factoryInitialize' method directly after the non-default
  // constructor call, namely 'E(const std::string, Record&)'
  // where 'E' is the templated sub-entity -- see 'register.cc'
  // for the actual code.

  virtual
  void                                      // called by factory after construction
  factoryInitialize();

  // ACCESSORS

  const std::string
  getIdentifier() const;                    // returns d_identifier

  const std::string                         // primarily for use in ctor and dtor logging
  getIdAndKind                              // returns d_identifier and if link entity
  (const std::string& auxMsg = "") const;   // optional auxillary message appended in []

  const std::string                         // provided for testing
  getTypeStr() const;                       // RTTI name, duly cleaned for g++

  const int
  getUseCount() const;                      // includes copy in the respective census

  const std::string
  report
  (const std::string msg = "(not given)") const;

  // INSTANCE MANIPULATORS -- for the set of 'EntityFactory' '::createType<>' functions

  template <typename E>                     // CAUTION: must be defined in header file
  void
  processFabricatedPtr
  (shared_ptr<E> me)                        // called with implicit instantiation
  {
    s_censusFull.push_back(me);             // copy into shared census

    std::string compilerName = typeid(E).name();  // 'name' returns 'const char*'
    d_typeStr = xeona::demangle(compilerName);    // cleaning substantive only for g++
    s_logger->repx(logga::adhc, "type string", getTypeStr());
  }

  // STATIC MANIPULATORS -- for the "lazy" linking process

  // ----------------------------------------------------------
  //  MEMBER FUNCTION : makeLinkEntity <> (static)
  // ----------------------------------------------------------

  // CAUTION: this function is defined in the header file to
  // allow for implicit template instantiation

  template <typename E>
  static
  assign_ptr<E>
```

```
  makeLinkEntity                                // static factory function
  (const std::string& soughtId)                 // note sought id rather than own identifier
  {
    s_logger->repx(logga::dbug, "entering member function", soughtId);

    assign_ptr<E> link(new E(soughtId));        // "factory" call (no registration required)
    s_censusLink.push_back(link);               // upcast copy constructor called
    return link;
  }

  static
  bool
  linkAll();                                    // CAUTION: must follow entity construction

  // STATIC ACCESSORS -- general use

  static
  int                                           // return "full" census size
  getFullPopn();                                // static member funcs cannot be cv-qualified

  static
  const int                                     // unitless
  getHorizonSteps();

  static
  const int                                     // time in seconds
  getHorizonInterval();

  static
  const int
  getHorizonStartHour();

  static
  const int
  getHorizonStartDay();

  static
  const xeona::Hemisphere
  getHorizonHemisphere();

  static
  const bool                                    // 'true' if in "full" census, else 'false'
  confirmIdentifier
  (const std::string& soughtId);

  static
  shared_ptr<Overseer>
  retOverseer();                                // could equally classify as a manipulator

  static
  std::vector<shared_ptr<DomainController> >
  retDomains();                                 // could equally classify as a manipulator

  // STATIC ACCESSORS FOR TEST PURPOSES

  static
  int                                           // the number of (formed or unformed) links
  getLinkPopn();

  static
  int                                           // return "raw" census size, links and fulls
  getRawPopn();

  static
  const std::list<const Entity*>&               // return a 'std::list' reference
  getCensusRaw();

  static
  std::string                                   // return formatted identifiers
  traverseFullPopulation();                     // uses the "full" census

  // STATIC MANIPULATORS

  static
  bool                                          // 'true' unless call was inappropriate
  setHorizonSteps
  (const int steps);

  static
  void
  addWant(const std::string& entityType);       // append to list of mandatory entities
```

```
  static
  void
  checkWants
  (std::vector<std::string>& buffer);        // confirm that all mandatory entities exist

  // STATIC MANIPULATORS FOR TEST PURPOSES

#ifdef _XUTEST                               // a compile-time failure under normal build

  static
  void
  setHorizonDirectly
  (const int steps,
   const int interval)
  {
    s_horizonSteps    = steps;               // unitless
    s_horizonInterval = interval;            // in seconds
  }

#endif // _XUTEST

  // INSTANCE UTILITY FUNCTIONS

protected:

  bool                                       // 'true' indicates unique
  isUniqueIdentifierFull();                  // "full" means the full census is used

private:

  const shared_ptr<Entity>
  retConstMe() const;                        // wrapper call to 'retConstSharedPtr'

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : retFull <>
  // -----------------------------------------------------------
  //
  //  Design notes
  //
  //      This function can only return a "full" version.
  //
  // -----------------------------------------------------------

protected:

  template <typename E>
  shared_ptr<E>
  retFull() const
  {
    const std::string identifier = getIdentifier();
    shared_ptr<E> temp = dynamic_pointer_cast<E>(Entity::retSharedPtr(identifier));
    if ( temp == 0 )
      {
        const std::string etype = xeona::demangle(typeid(E).name());
        const std::string mtype = getTypeStr();
        s_logger->repx(logga::warn, "dynamic pointer cast failed", mtype);
        if ( xeona::nopro == false )         // meaning option '--krazy' not applied
          {
            s_logger->repx(logga::dbug, "will throw xeona::full_link_fail", "");
            throw xeona::full_link_fail(identifier,    // identifier
                                        etype,         // template type
                                        mtype);        // my type
          }
      }
    return temp;
  }

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : retMe
  // -----------------------------------------------------------
  //
  //  Design notes
  //
  //      The <Entity const> means that the controlled resource
  //      cannot be switched.
  //
  //      This function can return both "full" and "link"
  //      versions, depending on the host.
  //
  // -----------------------------------------------------------
```

```
  shared_ptr<Entity const>
  retMe() const
  {
    return retMe<Entity const>();
  }

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : retMe <>
  // -----------------------------------------------------------
  //
  //  Design notes
  //
  //      The <E const> means that the controlled resource cannot
  //      be switched.
  //
  //      This function can return both "full" and "link"
  //      versions, depending on the host.
  //
  //      See also: Boost mailing list archive: From: Raffaele
  //      Romito / Date: 2006-01-10 10:58:23 / search-terms:
  //      "shared_ptr retrieve inherit base"
  //
  // -----------------------------------------------------------

  template <typename E>
  shared_ptr<E const>
  retMe() const
  {
    shared_ptr<E const> temp = dynamic_pointer_cast<E const>(shared_from_this());
    if ( temp == 0 )
      {
        const std::string type = xeona::demangle(typeid(E).name());
        s_logger->repx(logga::warn, "dynamic pointer cast failed", type);
      }
    return temp;
  }

  // STATIC UTILITY FUNCTIONS

private:                                    // CAUTION: note the friendships

  static
  const shared_ptr<Entity>
  retConstSharedPtr                         // wrapper call to 'retSharedPtr'
  (const std::string& soughtId);

  static
  shared_ptr<Entity>
  retSharedPtr                              // could equally classify as a manipulator
  (const std::string& soughtId);

  static
  bool                                      // 'true' if all "wanted entities" were got
  checkEntityTypesLists();

  // STREAM INSERTION SUPPORT

public:

  virtual                                   // CAUTION: needed for polymorphic behavior
  std::ostream&
  streamOut                                 // support for overloaded operator<<
  (std::ostream& os) const;

  // INSTANCE DATA

private:

  const std::string    d_identifier;        // unique identifier as given in model
  const Record&        d_record;            // used passively for housekeeping tasks
  std::string          d_typeStr;           // based on RTTI (run-time type information)

  // STATIC DATA

private:

  static int                  s_horizonSteps;          // unitless
  static int                  s_horizonInterval;       // in seconds
  static int                  s_horizonStartHour;      // [0,23]
  static int                  s_horizonStartDay;       // [1,365]
```

```
  static xeona::Hemisphere    s_horizonHemisphere;     // {e_north, e_south}

  static int                  s_linkAllCallCount;      // 'linkAll' calls count

  static std::list<const Entity*>          s_censusRaw;     // all Entity instances
  static std::vector<shared_ptr<Entity> >  s_censusFull;    // "full" Entity census
  static std::vector<assign_ptr<Entity> >  s_censusLink;    // "link" Entity census
  static std::vector<std::string>          s_identsLink;    // parallel sought identifiers

  static std::vector<std::string>    s_entityTypesWant;     // list of mandatory types
  static std::vector<std::string>    s_entityTypesGot;      // list of types supplied
  static std::vector<std::string>    s_entityTypesDiff;     // list of types not got

  static LinkLogger                  s_linkLogger;          // monitor linking outcomes

protected:

  static logga::spLogger     s_logger;            // shared_ptr to single logger object

  // stand-in quantites for use in link entity constructors and similar

  static Record         s_recordStandIn;
  static std::string    s_stringEmpty;
  static double         s_doubleZero;
  static int            s_intZero;
  static bool           s_boolFalse;

  // UNIT TESTING METHODS -- defined here too for convenience

public:

#ifdef _XUTEST                              // specifically for class 'Datfact_UT'

  // CAUTION: must be public but DO NOT place the access
  // specifier within this preprocessor 'hash-ifdef'

  virtual
  double
  getX() const
  {
    return std::numeric_limits<double>::quiet_NaN();   // output streams as a "nan"
  }

  virtual
  void
  multiplyX(const double factor)
  { }

#endif // _XUTEST

}; // class 'Entity'

// ----------------------------------------------------------
//  FREE FUNCTION   : operator<< (std::ostream&, Entity&)
// ----------------------------------------------------------

// CAUTION: place after client class declaration

inline                                      // CAUTION: inline essential
std::ostream&
operator<<                                  // overloaded operator
(std::ostream& os,
const Entity&  en)
{
  return en.streamOut(os);                  // wrapper to streamOut member function
}

// ----------------------------------------------------------
//  CLASS           : FullEntity (abstract)
// ----------------------------------------------------------
//  Description  : stepping stone sub-entity
//  Role         : part of the full entity hierarchy
//  Techniques   : inheritance
//  Status       : complete
// ----------------------------------------------------------

class FullEntity
  : public Entity
{
  // DISABLED
```

```
private:

  FullEntity();                                          // zero-argument constructor
  FullEntity(const FullEntity& orig);                    // copy constructor
  FullEntity& operator= (const FullEntity& orig);        // copy assignment operator

  // CREATORS

public:

  explicit
  FullEntity(const std::string entityId) :
    Entity(entityId),
    d_builtinRemark(s_stringEmpty)
  { }

  explicit                                    // disable implicit type conversions (string)
  FullEntity
  (const std::string entityId,
   Record&           record);

  virtual
  ~FullEntity() = 0;                          // create abstract class

  // MANIPULATORS

  virtual
  void                                        // called by factory after construction
  factoryInitialize();

  // STATIC ACCESSORS

  template<typename E>                        // downcasting employed
  static
  const int                                   // returns number of elements in 'vec'
  listToVec                                   // explicit multiple entity linking
  (const std::string&           list,         // order is significant
   std::vector<shared_ptr<E> >& vec);         // pass by non-const reference, note the 'E'

  // INTERNAL DATA

protected:

  std::string&    d_builtinRemark;            // normally re-assigned in derived classes

private:

  static int      s_fullCount;                // "full" entity con/destructor call count

}; // class 'FullEntity'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::putxId
// ----------------------------------------------------------
//  Description  : conveniently add 'putx' output
//  Role         : called in various processing loops to improve logging readability
//  Techniques   : 'boost::format', static variables
//  Status       : complete
// ----------------------------------------------------------

namespace xeona
{
  int                                         // empty/null entities count passed to date
  putxId
  (const shared_ptr<Entity> e,                // full or link entity
   const std::string&       comment = "");   // optional comment

} // namespace 'xeona'

#endif // _ENTITY_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : gateway.h
//  file-create-date : Thu 23-Oct-2008 11:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : gateway entity which span commitment domains / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/gate.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Gateway-related notation
//
//      To be read in conjuction with unit 'optgate'.
//
//      CAUTION: incomplete and incorrect
//
//          side   terminology   OSP   comment
//          ------------------------------------------
//          buy
//          buy    saturation     -
//          buy    tariff        taf
//          sel
//          sel    obligation    qan
//          sel    offer          -
//          ------------------------------------------
//          abbreviations:
//          OSP = optimization sub-problem
//          ops = operations, ots = obligation, qan = quantity, taf = tariff
//
//  Lead-up
//
//      Lead-up development in 'frag-xeona-gateway-6.cc'.
//
//  Diamond inheritance
//
//      Overview
//
//          The design of the gateway entity make use of a
//          diamond inheritance graph.  This necessitates, in
//          this case, virtual inheritance and thereby introduces
//          some additional considerations, including base class
//          initialization.
//
//          See Stroustrup (1997 p396-397) for a similar 'Radio'
//          example which demonstrates "diamond-shaped
//          inheritance".
//
//      Virtual base classes
//
//          Some comments from Lischner (2003):
//
//              "When constructing a class that has virtual
```

```
//              base classes, the most-derived class's
//              constructors must initialize all the virtual
//              base classes.  If a constructor does not
//              explicitly initialize a virtual base class, the
//              virtual base class's default [zero-argument]
//              constructor is used.  Initializers for the
//              virtual base classes are ignored when all base
//              class constructors run."  (p165)
//
//              "Warning: You should design your virtual base
//              classes so they require only the default
//              constructor.  Otherwise, you impose a burden on
//              every derived class to initialize the virtual
//              base properly.  Any change to the parameters of
//              the virtual base constructor necessitates a
//              corresponding change to ever constructor for
//              every class that derives, directly or indirectly,
//              from the virtual base class."  (p166)
//
//          Indeed, Lischner's advice has NOT been taken and the
//          aforementioned burden applies.  That said, the base
//          class constructor will be (in the absence of
//          catastrophes) completely stable.
//
//      Prior code
//
//          From r2197 to r2000, two gateway implementations were
//          supported.  Refer to r2200 for the original basic
//          gateway implementation (note also my alias 'svcatr'):
//
//              $ svn cat --revision 2200 b/gateway.h  | less
//              $ svn cat --revision 2200 b/gateway.cc | less
//
//          See 'frag-diamond-inheritance-2.cc' for background
//          fragment code development in relation to "diamond"
//          inheritance.
//
//      References
//
//          Lischner, Ray.  2003.  C++ in a nutshell : a language
//            and library reference.  O'Reilly and Associates,
//            Sebastopol, California, USA.  ISBN 0-596-00298-X.
//
//          Stroustrup, Bjarne.  1997.  The C++ programming
//            language -- Third edition.  Addison-Wesley, Reading,
//            Massachusetts, USA.  ISBN-10 0-201-88954-4.

//  HEADER GUARD

#ifndef _GATE_H_
#define _GATE_H_

//  AD-HOC NOTES
//
//  Commodities class list
//
//      CmOxidize
//      CmCarbonCert
//      CmCarbonSeq
//      CmElectricity
//      CmWork
//      CmHeat
//      CmThermalFluid
//      CmFunds
//      CmProductiveLand (limited support)
//
//  The now abandoned typedef examples were added in commit
//  r2193, namely:
//
//      typedef GateStatedTariff<CmWork> GateStatedTariffWork;
//
//  LOCAL AND SYSTEM INCLUDES

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../b/tictoc.h"      // inherited interface for entities using common calls
#include "../b/costreg.h"     // cost registers
#include "../b/block.h"       // block entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
```

```
#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <utility>            // STL pair, make_pair()

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // 'Gateway' constructor argument
class DomainController;                       // member function usage and member data

class OfferOsp;                               // 'BuySide' and 'Selgate' member data
class OperationsOsp;                          // 'BuySide' and 'Selgate' member data
class QuantityOsp;                            // 'BuySide' member data

template <typename C> class Cable;            // 'GateCom' member data
template <typename C> class Socket;           // 'GateCom' member data

//  CODE

// ------------------------------------------------------------
//  CLASS           : Gateway (abstract)
// ------------------------------------------------------------
//  Description  : gateway base class
//  Role         : step in the entity inheritance web
//  Techniques   : virtual multiple inheritance, gateway-specific abstract functions
//  Status       : mostly complete
//
//  Ad-hoc thoughts
//
//      gates only know financial costs
//      obj's are set iff 'e_shortrunFin' is set, else zeroed
//
//      recover costs calls are needed
//
//      The fact that gates only know financial costs is in
//      keeping with the information destroying characteristic of
//      commodity markets.
//
//      HANG ON .. gray costs should be supported
//
//  Coding
//
//      The data members here are common, occur only once, and do
//      not require scope resolution to resolve ambiguity (due to
//      virtual inheritance).
//
// ------------------------------------------------------------

class Gateway :                               // the bottom of gateway tree
  public Block
{

  // TYPEDEFS

public:

  typedef std::pair<double,                   // lower bound
                    double                    // upper bound
                    >bounded_type;

  // DISABLED

private:

  Gateway();                                  // zero-argument constructor
  Gateway(const Gateway& orig);               // copy constructor
  Gateway& operator= (const Gateway& orig);   // copy assignment operator

  // CREATORS

public:

  explicit
  Gateway
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum);      // passed thru to 'TicToc' constructor

  virtual ~Gateway() = 0;                     // create abstract class

public:
```

```
  // ACCESSORS

  // boolean label status for DFS and CTA graph algorithms

  bool getTilde() const;                    // depth-first search label
  bool getTheta() const;                    // capacity back-track lock
  bool getStar()  const;                    // transaction forward-track lock
  bool getHash()  const;                    // fully-capacitated flag

  // sale status

  bool   getTransacted()  const;            // 'true' if "sold"
  double getTransaction() const;
  double getTotalCost()   const;

  // navigation calls for DFS and CTA graph algorithms

  virtual
  shared_ptr<DomainController>              // my partner domain controller
  hop
  (const shared_ptr<DomainController> me) = 0;

  // obtain associated blocks

  virtual
  shared_ptr<Block>
  getDemander() const = 0;                  // obtain demander -- the socket-side block

  virtual
  shared_ptr<Block>
  getSupplier() const = 0;                  // obtain supplier -- the cable-side block

  // MANIPULATORS

  void unsetTransaction();                  // hop-relitigation extension only

  // calls used by 'DomainController::capset' function

  void setLowerCapacity(const double lowerCta);
  void setUpperCapacity(const double upperCta);

  // flag calls for DFS and CTA graph algorithms -- these
  // functions also return their previous status

  bool markTilde();
  bool unmarkTilde();
  bool markTheta();
  bool unmarkTheta();
  bool markStar();
  bool unmarkStar();
  bool markHash();
  bool unmarkHash();

  // reset call for DFS and CTA graph algorithms

  void reset();

protected:

  // these calls are wrapped by normal 'initialize' calls, with
  // the latter defined in the next layer of specialization

  virtual void initializeBuySide (const int step, shared_ptr<svif::SolverIf> solver) = 0;
  virtual void initializeSelSide (const int step, shared_ptr<svif::SolverIf> solver) = 0;

  // these calls are wrapped by normal 'constrain' calls, with
  // the latter defined in the next layer of specialization

  virtual const int constrainBuySide(const xeona::DomainMode capacityMode) = 0;
  virtual const int constrainSelSide(const xeona::DomainMode capacityMode) = 0;

  // these calls are necessarily redefined in the various
  // concrete gateways

  virtual void washupBuySide() = 0;
  virtual void washupSelSide() = 0;

  // these calls are redefined in the 'BuySide' and 'SelSide'

  virtual void loadRegisterBuySide(const int step) = 0;
  virtual void loadRegisterSelSide(const int step) = 0;
```

```
  // INSTANCE DATA

protected:

  // the following may not be necessary or even convenient

  bounded_type           d_techCapacity;     // technical (lower, upper) capacity
  bounded_type           d_commCapacity;     // commercial (lower, upper) capacity

  // canonical cross-domain data (derived data such as marginal
  // price and total payment falls under the responsibility of
  // the concrete gateway class)

  bounded_type           d_capacity;         // sale (lower, upper) capacity
  double                 d_transaction;      // sale quantity
  double                 d_totalCost;        // cost of transaction

  int                    d_dutyGol;          // transaction global col

private:

  // graph algorithm boolean labels

  bool                   d_tilde;            // depth-first search label
  bool                   d_theta;            // capacity back-track lock
  bool                   d_star;             // transaction forward-track lock
  bool                   d_hash;             // fully-capacitated flag

  // STATIC DATA

protected:

  // for convenience

  static const double    s_inf;              // floating point infinity
  static const double    s_nil;              // nonsensical at NaN (see implementation)

}; // class 'Gateway'

// ------------------------------------------------------------
//  CLASS           : BuySide (abstract)
// ------------------------------------------------------------
//  Description  : add in buy-side functionality
//  Role         : step in the entity inheritance web
//  Techniques   : brings in the 'TicToc' interface and provides definitions
//  Status       : mostly complete
//
//  Coding
//
//      Add dedicated buy-side members here -- these should be
//      unique by design (or introduced one layer down).
//
// ------------------------------------------------------------

class BuySide :
  public virtual Gateway,
  public TicToc,
  public CostRegister
{
  // CREATORS

public:

  explicit
  BuySide
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum);

  virtual ~BuySide() = 0;

  // MANIPULATORS

public:

  void
  registerDomain
  (const std::string& actorId);

  void
```

```
  registerDomain
  (shared_ptr<DomainController> domcon);

  virtual
  void
  initialize
  (const int                  step,
   shared_ptr<svif::SolverIf> solver);       // solver instance passed thru

  virtual
  const int
  constrain                                 // set constraints
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                 // calls 'washupBuySide'

  // store and unset calls for CTA graph algorithm

  double recordTransaction();               // also returns transaction for convenience

  // INSTANCE DATA

protected:

  shared_ptr<DomainController>    d_controller;   // controlling domain for this side

}; // class 'BuySide'

// ----------------------------------------------------------
//  CLASS           : SelSide (abstract)
// ----------------------------------------------------------
//  Description  : add in sel-side functionality
//  Role         : step in the entity inheritance web
//  Techniques   : brings in the 'TicToc' interface and provides definitions
//  Status       : mostly complete
//
//  Coding
//
//      Add dedicated sel-side members here -- these should be
//      unique by design (or declared in 'Gateway').
//
// ----------------------------------------------------------

class SelSide :
  public virtual Gateway,
  public TicToc,
  public CostRegister
{
  // CREATORS

public:

  explicit
  SelSide
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum);

  virtual ~SelSide() = 0;

  // MANIPULATORS

public:

  void
  registerDomain
  (const std::string& actorId);

  void
  registerDomain
  (shared_ptr<DomainController> domcon);

  virtual
  void
  initialize
  (const int                  step,
   shared_ptr<svif::SolverIf> solver);       // solver instance passed thru

  virtual
```

```
  const int
  constrain                                 // set constraints
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                 // calls 'washupSelSide'

  // INSTANCE DATA

protected:

  shared_ptr<DomainController>    d_controller;   // controlling domain for this side

}; // class 'SelSide'

// ----------------------------------------------------------
//  CLASS           : GateCom <> (abstract)
// ----------------------------------------------------------
//  Description  : single templated gateway class
//  Role         : for use by concrete gateway classes
//  Techniques   : templated on 'C' for 'Commodity'
//  Status       : mostly complete
//
//  Coding
//
//      Add undifferentiated single members here -- these are now
//      also clear of the multiple virtual ('TicToc' excepted)
//      inheritance web.
//
//      The 'cable' and 'socket' are defined here.
//
// ----------------------------------------------------------

class BandedTariffSet;

template <typename C>
class GateCom :
  public BuySide,                           // CAUTION: virtual inheritance not wanted
  public SelSide                            // CAUTION: virtual inheritance not wanted
{
  // CREATORS

public:

  explicit
  GateCom
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum);

  virtual ~GateCom() = 0;

public:

  // FOR UNIT TESTING

  void unitTestSay();                       // strictly for testing

  // ACCESSORS

  virtual
  shared_ptr<DomainController>              // my partner domain controller
  hop
  (const shared_ptr<DomainController> me);

  virtual
  shared_ptr<Block>
  getDemander() const;                      // obtain demander -- the socket-side block

  virtual
  shared_ptr<Block>
  getSupplier() const;                      // obtain supplier -- the cable-side block

  virtual
  shared_ptr<BandedTariffSet>
  obtainTariffSet() const = 0;

  // MANIPULATORS

  // CAUTION: the note below for 'constrain' may apply here also
```

```
  // -- but this requirement was not expressly checked for
  // 'initialize' -- alternatively the caution may be stale in
  // both cases (which I suspect it might be)

  virtual
  void
  initialize
  (const int                  step,
   shared_ptr<svif::SolverIf> solver)        // solver instance passed thru
  {
    s_logger->repx(logga::warn, "use upcast call instead", "");
  }

  // CAUTION: the compiler requires the call diamond to be
  // completed, hence this next 'constrain' declaration, also
  // defined here so no template instantiations

  virtual
  const int
  constrain
  (const xeona::DomainMode capacityMode)
  {
    s_logger->repx(logga::warn, "use upcast call instead", "");
    return -2;                                // nonsensical return
  }

  virtual
  void
  washup()
  {
    s_logger->repx(logga::warn, "use upcast call instead", "");
  }

  // INSTANCE DATA

protected:

  // tied quantities

  shared_ptr<Cable<C> >     d_cable;
  shared_ptr<Socket<C> >    d_socket;

//shared_ptr<std::vector<double> >    d_lowerTechCapacitys; // determined during solution
//shared_ptr<std::vector<double> >    d_upperTechCapacitys; // determined during solution

  // local quantities

}; // class 'GateCom<>'

#endif // _GATE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : gate1.h
//  file-create-date : Wed 15-Apr-2009 21:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete gateways 1 - stated tariff / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/gate01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _GATE01_H_
#define _GATE01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/gate.h"        // gateway entity

#include "../b/tictoc.h"      // inherited interface for entities using common calls
#include "../b/block.h"       // block entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class BandedTariffSet;                      // from "b/bandtaf.h" with banded tariff set

//  CODE

// ------------------------------------------------------------
//  CLASS           : GateStatedTariff <>
// ------------------------------------------------------------
//  Description  : prescribed tariffset with embedded information
//  Role         : concrete class (and the first gateway implemented)
//  Techniques   : the 'optgate' OSP will later support concave tariffsets
//  Status       : somewhat complete
//
//  Design notes
//
//      Intent
//
//          In this class, the tariffset is prescribed -- meaning
//          that it is not stochastic, not unilaterally set by
//          one or other party, and not jointly renegotiated at
//          each interval.  Indeed, the modeler can be
//          interpreted as, in some sense at least, the broker of
//          this contract.
```

```
//
//      Commitment mode sum
//
//          The commitment mode sum MUST be specified as a
//          default constructor argument.
//
//      The 'using declaration'
//
//          First see Stroustrup (1997 pp392-394).  In this case,
//          the "protected" access specifier restricts external
//          visibility.
//
//     The so-called in-side tariffset structure
//
//         Data member is 'd_ofr' is shared between the buy and
//         sell sides.  Hence the designation "in" side.
//
//  CAUTION: function 'initialize'
//
//      Function 'initialize' does not need redefinition here but
//      note the need to resolve the call (see the unit test for
//      a working example), namely:
//
//          buygate->BuySide::initialize(step, solver);
//
//  CAUTION: "object missing" error
//
//      This can be solved by using a correct scope resolution,
//      for example:
//
//          std::cout << GateCom<C>::SelSide::d_commitmentMode;
//
//      or alternatively (absolutely not recommended):
//
//          std::cout << this->SelSide::d_commitmentMode;
//
//      to avoid the following compiler complaint:
//
//          "error: object missing in reference to 'TicToc::d_commitmentMode'"
//
// ------------------------------------------------------

//  FORWARD (PARTIAL) DECLARATIONS

class OfrTariffSet_A;                       // CAUTION: for class declaration typedef
class QanObligToSupply_A;                   // CAUTION: for class declaration typedef
class QanTechCapacity_A;                    // CAUTION: for class declaration typedef

//  CODE

template <typename C>
class GateStatedTariff :
  public GateCom<C>
{
  // USING DECLARATIONS

protected:

  // CAUTION: the following place variables in common scope for
  // this class, thereby remove a "declaration must be available"
  // or "not declared in this scope" errors

  using Entity::s_logger;
  using Entity::getIdentifier;
  using Entity::getIdAndKind;

  using Block::d_dutyStats;
  using Block::d_sizeStats;

  using Gateway::d_techCapacity;
  using Gateway::d_capacity;
  using Gateway::d_transaction;

  using GateCom<C>::d_cable;
  using GateCom<C>::d_socket;

  // TYPEDEFS

protected:

  typedef OfrTariffSet_A     OfrTariffSet;        // used for switching implementations
```

```
  typedef QanObligToSupply_A QanObligToSupply;    // used for switching implementations
  typedef QanTechCapacity_A  QanTechCapacity;     // used for switching implementations

  // CREATORS

public:

  explicit
  GateStatedTariff
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum = xeona::e_commitmentModes);

  virtual
  ~GateStatedTariff();

  // ACCESSORS

  virtual
  shared_ptr<BandedTariffSet>
  obtainTariffSet() const
  {
    // intitial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    // return, noting that the tariff set is not "locked" and
    // could be modified by external objects -- that said,
    // function 'BandedTariffSet::getMarginalPrice' is 'const'
    return d_tariffset;
  }

  // MANIPULATORS

public:

  // from class 'TicToc'

  virtual void establish();                  // necessary redefinition, currently hollow
  virtual void conclude();                   // necessary redefinition, currently hollow

  // from class 'Gateway'

  virtual void initializeBuySide (const int step, shared_ptr<svif::SolverIf> solver);
  virtual void initializeSelSide (const int step, shared_ptr<svif::SolverIf> solver);

  virtual const int constrainBuySide(const xeona::DomainMode capacityMode);
  virtual const int constrainSelSide(const xeona::DomainMode capacityMode);

  virtual void washupBuySide();
  virtual void washupSelSide();

  virtual void loadRegisterBuySide(const int step);
  virtual void loadRegisterSelSide(const int step);

  // INSTANCE DATA

private:

  // tied quantities

  shared_ptr<std::vector<std::string> >    d_tariffsets;
  shared_ptr<std::vector<double> >         d_definedCapacitys;   // [1]

  // [1] normally invariant -- but the technical capacity could
  // be made context-dependent in a derived class if need be

  shared_ptr<std::vector<double> >         d_quantitys;
  shared_ptr<std::vector<double> >         d_marginalPrices;
  shared_ptr<std::vector<double> >         d_totalCosts;

  // internal quantities

  shared_ptr<BandedTariffSet>              d_tariffset;     // filled using 'pushString'

  // OSP specializations required

  shared_ptr<QanTechCapacity>              d_capSel;        // sel-side capacity
  shared_ptr<QanTechCapacity>              d_capBuy;        // buy-side capacity
  shared_ptr<QanObligToSupply>             d_ots;           // sel-side obligation
  shared_ptr<OfrTariffSet>                 d_ofr;           // in-side tariffset
```

```
};

//  ==== XEDOC ================================================
//
//  entity.gate-stated-tariff-*-0
//
//      class                                   > GateStatedTariff:Work
//
//        this gateway entity used fixed (rather than stochastic,
//        unilaterally set, or negotiated) tariffs
//
//        the 'Cm*' qualifier (here * = Work) specifies the
//        relevant high-level commodity class
//
//        my socket label is 'sock-1'
//
//      builtin-remark s                        <
//
//      socket l                                > "teas-1.work-1"
//      common-commodity l                      > "cm-work-0"
//
//        the above fields define my supplier and their socket
//        label and our common commodity
//
//      tariffsets X                            > "4.4 * 40.00e+06 28.00e-09" ..
//
//        the optional leading term (4.4) in the tariffsets field
//        represents a fixed charge [$/s] -- the remaining
//        star-separated pairs being not necessarily convex
//        quantity/price [J $] offers
//
//      defined-capacitys [W] F                 > 50.00e+06 ..
//
//        data capture
//
//      quantitys [*] F                         < 0.0 ..
//      marginal-prices [$/*] F                 < 0.0 ..
//      total-costs [$] F                       < 0.0 ..
//
//  ============================================================

#endif // _GATE01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : junc.h
//  file-create-date : Mon 26-Oct-2009 12:44 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : demand split/join junction entity / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/junc.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _JUNC_H_
#define _JUNC_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/tictoc.h"      // inherited interface for entities using common calls
#include "../b/block.h"       // block entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ---------------------------------------------------------
//  CLASS           : DemandJunction (abstract)
// ---------------------------------------------------------
//  Description  : junction base class
//  Role         : step in the inheritance web (intentionally sidesteps 'TechnicalAsset')
//  Techniques   : inheritance
//  Status       : complete
//
//  Design notes
//
//      Constructor
//
//          Unlike 'TechnicalAsset', this class constructor does
//          not take the following third argument: const int
//          commitmentModeSum.  That instead is hardcoded in the
//          constructor initialization list, namely:
//
//              xeona::e_commitmentModes
//
//      Explicit listing
//
//          These entities are explicitly listed in the domain
//          controller XEM record, namely:
//
```

```
//              demand-junctions L > "teas-demand-2-split-0"
//
//          The underlying reason for this approach was to
//          maintain a clean design during development.
//
//      Alternative code for automatic location (suggestion)
//
//          These entities could, perhaps, remain implicit and be
//          located under the appropriate 'restructure' call -
//          noting that connections are fixed prior to the first
//          'restructure' call and that the relevant solver is
//          also known to that call.
//
//          Therefore it may be possible to locate junctions
//          automatically.  Moreover, as junctions do not form
//          costs, they need not be part of the entity management
//          hierarchy.
//
//          Possibly, add near end of 'Interface::bindOsp':
//
//          const bool ret                  = d_cnn->bindGols(...);
//          shared_ptr<Entity> part         = getPartner();
//          shared_ptr<DemandJunction> junc = dynamic_pointer_cast<DemandJunction>(part);
//          if ( junc ) junc->constrain(capacityMode);
//
// ------------------------------------------------------------

class DemandJunction :
  public Block,
  public TicToc
{
  // DISABLED

private:

  DemandJunction();                                          // zero-argument constructor
  DemandJunction(const DemandJunction& orig);                // copy constructor
  DemandJunction& operator= (const DemandJunction& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  DemandJunction
  (const std::string entityId,
   Record&           record);

  virtual
  ~DemandJunction() = 0;                     // create abstract class

  // CALLS

public:

  virtual void establish() { }               // necessary redefinition
  virtual void conclude()  { }               // necessary redefinition

}; // class 'DemandJunction'

#endif // _JUNC_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : junc01.h
//  file-create-date : Mon 26-Oct-2009 12:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete demand split junctions / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/junc01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _JUNC01_H_
#define _JUNC01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/junc.h"        // demand split/join junction entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

//  CODE

// ----------------------------------------------------------
//  CLASS           : JuncDemand2Split <>
// ----------------------------------------------------------
//  Description  : a two cable junction
//  Role         : demand splitting
//  Techniques   : class template, 'JncSplit2'
//  Status       : complete
//
//  Design notes
//
//      The various typename 'C' template specializations take
//      place in 'b/register.cc', more specifically within the
//      EFIRE macro statements.
//
//      With regard to the OSP class naming convention, a
//      trailing '_X' is not required or used.  That is because
//      there should be no need to develop different versions.
//
```

```
// ----------------------------------------------------------

class JncSplit2;                              // forward (partial) declaration

template <typename C>                         // 'C' for commodity
class JuncDemand2Split :
  public DemandJunction
{
  // DISABLED

private:

  JuncDemand2Split();                                          // zero-argument ctor
  JuncDemand2Split(const JuncDemand2Split& orig);              // copy constructor
  JuncDemand2Split& operator= (const JuncDemand2Split& orig);  // copy assignment operator

  // CREATORS

public:

  explicit
  JuncDemand2Split
  (const std::string entityId,
   Record&           record);

  virtual
  ~JuncDemand2Split();

  // CALLS

public:

  virtual
  const int                                   // returns zero (no duty coupling)
  constrain                                   // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                   // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_junctionCommodity;  // for the 'create' interface calls
  shared_ptr<Cable<C> >     d_cable1;
  shared_ptr<Cable<C> >     d_cable2;
  shared_ptr<Socket<C> >    d_socket;

  // local quantities

  shared_ptr<JncSplit2>     d_ops;            // specialization required

}; // class 'JuncDemand2Split<>'

//  ==== XEDOC ==================================================
//
//  entity.junc-demand-2-split-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                   > JuncDemand2Split:Elec
//
//        a two cable demand splitting junction for given commodity
//
//        my sole socket label is 'sock-1'
//
//      builtin-remark s                        <
//
//      socket-1 l                              > "teas-supplier-1.elec-1"
//      socket-2 l                              > "teas-supplier-2.elec-1"
//
//        socket-1 and socket-2 are my two potential suppliers
//
//      junction-commodity l                    > "cm-electricity-0"
//
//        junction-commodity defines the underlying commodity
```

```
//
//  ============================================================

#endif // _JUNC01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : junc02.h
//  file-create-date : Tue 27-Oct-2009 09:17 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete demand join junctions / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/junc02.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _JUNC02_H_
#define _JUNC02_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/junc.h"        // demand split/join junction entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

//  CODE

// ----------------------------------------------------------
//  CLASS           : JuncDemand2Join <>
// ----------------------------------------------------------
//  Description  : a two socket junction
//  Role         : demand joining
//  Techniques   : class template, 'JncJoin2'
//  Status       : complete
//
//  Design notes
//
//      The various typename 'C' template specializations take
//      place in 'b/register.cc', more specifically within the
//      EFIRE macro statements.
//
//      With regard to the OSP class naming convention, a
//      trailing '_X' is not required or used.  That is because
//      there should be no need to develop different versions.
//
```

```
// ----------------------------------------------------------

class JncJoin2;                                 // forward (partial) declaration

template <typename C>                           // 'C' for commodity
class JuncDemand2Join :
  public DemandJunction
{
  // DISABLED

private:

  JuncDemand2Join();                                          // zero-argument ctor
  JuncDemand2Join(const JuncDemand2Join& orig);              // copy constructor
  JuncDemand2Join& operator= (const JuncDemand2Join& orig);  // copy assignment operator

  // CREATORS

public:

  explicit
  JuncDemand2Join
  (const std::string entityId,
   Record&           record);

  virtual
  ~JuncDemand2Join();

  // CALLS

public:

  virtual
  const int                                     // returns zero (no duty coupling)
  constrain                                     // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                     // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_junctionCommodity;  // for the 'create' interface calls
  shared_ptr<Cable<C> >     d_cable;
  shared_ptr<Socket<C> >    d_socket1;
  shared_ptr<Socket<C> >    d_socket2;

  // local quantities

  shared_ptr<JncJoin2>      d_ops;              // specialization required

}; // class 'JuncDemand2Join<>'

//  ==== XEDOC ==================================================
//
//  entity.junc-demand-2-join-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                     > JuncDemand2Join:Elec
//
//        a two socket demand joining junction for given commodity
//
//        my two socket labels are 'sock-1' and 'sock-2'
//
//      builtin-remark s                          <
//
//      socket-1 l                                > "teas-supplier-1.elec-1"
//
//        socket-1 is my sole supplier
//
//      junction-commodity l                      > "cm-electricity-0"
//
//        junction-commodity defines the underlying commodity
//
```

```
//  ============================================================

#endif // _JUNC02_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : lmpbid.h
//  file-create-date : Fri 17-Oct-2008 12:53 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : LMP auction bidset and support / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/lmpbid.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _LMPBID_H_
#define _LMPBID_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <utility>            // STL pair, make_pair()
#include <vector>             // STL sequence container

//  CODE

// ----------------------------------------------------------
//  CLASS           : LmpBidSet
// ----------------------------------------------------------
//  Description  : bidset abstraction for LMP (nodal) auctions
//  Role         : used to prepare, hold, and unpack a bidset
//  Techniques   : stand-alone class, 'std::pair', 'std::upper_bound'
//  Status       : complete and tested
// ----------------------------------------------------------

class LmpBidSet
{
  // TYPEDEFS

private:

  typedef std::pair<double, double> bid_type;     // namely (band, price)

  // FRIENDS

  friend class CtlLmpBid_A;                       // to access compiler-provided copy ctor

  // DISABLED

private:
```

```
  LmpBidSet(const LmpBidSet& orig);               // copy constructor
  LmpBidSet& operator= (const LmpBidSet& orig);   // copy assignment operator

  // CREATORS

public:

  LmpBidSet
  (std::string label = "");                  // optional label

  ~LmpBidSet();

  // UNARY OPERATORS

public:

  LmpBidSet& operator+= (const double& other);    // price increment
  LmpBidSet& operator-= (const double& other);    // price decrement
  LmpBidSet& operator*= (const double& other);    // price multiplier

  // MANIPULATORS

public:

  void
  clear();                                   // remove all elements

  int                                        // number of bids loaded this time
  pushString                                 // parses input, uses 'pushBid' to load data
  (const std::string sBidset);

  int                                        // current number of bids
  pushBid
  (const bid_type bid);

  void
  negate();                                  // multiply all unit prices by minus one

  void
  unnegate();                                // cancel any earlier 'negate' call

  bid_type
  popBig();                                  // pop biggest, meaning current highest price

  bid_type
  popSmall();                                // pop smallest, meaning current lowest price

  // ACCESSORS

  int                                        // current number of bids
  size() const;

  double
  getLoQuantity() const;

  double
  getHiQuantity() const;

  double
  getWeightedPrice() const;

  std::string                                // 'modelBidDelim'-separated for output
  stringify() const;

  std::string
  summarizeAll
  (const int indent) const;

  // INSTANCE DATA

private:

  std::string              d_label;          // optional
  std::vector<bid_type>    d_bidset;         // sorted
  bool                     d_negate;         // defaults to 'false'

  // STATIC DATA

private:

  static logga::spLogger    s_logger;        // shared_ptr to single logger object
```

```
};

#endif // _LMPBID_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : node.h
//  file-create-date : Tue 03-Nov-2009 12:18 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : LMP node entity / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/node.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _NODE_H_
#define _NODE_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/tictoc.h"      // inherited interface for entities using common calls
#include "../b/block.h"       // block entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ------------------------------------------------------------
//  CLASS           : LmpNode (abstract)
// ------------------------------------------------------------
//  Description  : LMP node base class
//  Role         : step in the inheritance web (intentionally sidesteps 'TechnicalAsset')
//  Techniques   : (nothing special)
//  Status       : more-or-less complete
//
//  Design notes
//
//      Similarities to class 'DemandJunction'
//
// ------------------------------------------------------------

class LmpNode :
  public Block,
  public TicToc
{
  // DISABLED

private:

  LmpNode();                                  // zero-argument constructor
  LmpNode(const LmpNode& orig);               // copy constructor
```

```
  LmpNode& operator= (const LmpNode& orig);  // copy assignment operator

  // CREATORS

public:

  explicit
  LmpNode
  (const std::string entityId,
   Record&           record);

  virtual
  ~LmpNode() = 0;                             // create abstract class

  // CALLS

public:

  virtual void establish() { }               // necessary redefinition
  virtual void conclude()  { }               // necessary redefinition

  // INSTANCE DATA

protected:

  // tied quantities

  shared_ptr<std::vector<double> >    d_nodalPrices;

}; // class 'LmpNode'

#endif // _NODE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : node01.h
//  file-create-date : Tue 03-Nov-2009 12:19 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete LMP nodes 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/node01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _NODE01_H_
#define _NODE01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/node.h"        // LMP node entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

//  CODE

// ------------------------------------------------------------
//  CLASS           : Node1Inj <>
// ------------------------------------------------------------
//  Description  : normal cable, bidirectional socket LMP node
//  Role         : spur line node with injection
//  Techniques   : class template, 'LmpCab1'
//  Status       : complete
//
//  Design notes
//
//      With regard to the OSP class naming convention, a
//      trailing '_X' is not required or used.  That is because
//      there should be no need to develop different versions.
//
// ------------------------------------------------------------

class LmpCab1;                                // forward (partial) declaration
```

```
template <typename C>                          // 'C' for commodity
class Node1Inj :
  public LmpNode
{
  // DISABLED

private:

  Node1Inj();                                     // zero-argument ctor
  Node1Inj(const Node1Inj& orig);                 // copy constructor
  Node1Inj& operator= (const Node1Inj& orig);     // copy assignment operator

  // CREATORS

public:

  explicit
  Node1Inj
  (const std::string entityId,
   Record&           record);

  virtual
  ~Node1Inj();

  // CALLS

public:

  virtual
  const int                                    // returns zero (no duty coupling)
  constrain                                    // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                    // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_nodeCommodity; // 'create' interface calls
  shared_ptr<Cable<C> >     d_cable;
  shared_ptr<Socket<C> >    d_socket;

  // local quantities

  shared_ptr<LmpCab1>       d_ops;           // specialization required

}; // class 'Node1Inj'

//  ==== XEDOC ================================================
//
//  entity.node-1-inj-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                    > Node1Inj:Elec
//
//        grid spur line node with injection
//
//        my sole bi-directional socket label is 'grid-1'
//
//      builtin-remark s                         <
//
//      socket-1 l                               > "teas-supplier-1.elec-1"
//
//        socket-1 is my normal supplier (injector)
//
//      node-commodity l                         > "cm-electricity-0"
//
//        node-commodity defines the underlying commodity
//
//      nodal-prices [$/J] F                     < 0.0 ..
//
//        to convert to the more usual [$/MWh], multiply the
//        given nodal-prices by 278
//
```

```
//  ============================================================

// -------------------------------------------------------
//  CLASS           : Node1Xit <>
// -------------------------------------------------------
//  Description  : bidirectional cable, normal socket LMP node
//  Role         : spur line node with exit
//  Techniques   : class template, 'LmpSoc1'
//  Status       : complete
//
//  Design notes
//
// -------------------------------------------------------

class LmpSoc1;                                  // forward (partial) declaration

template <typename C>                           // 'C' for commodity
class Node1Xit :
  public LmpNode
{
  // DISABLED

private:

  Node1Xit();                                       // zero-argument ctor
  Node1Xit(const Node1Xit& orig);                   // copy constructor
  Node1Xit& operator= (const Node1Xit& orig);       // copy assignment operator

  // CREATORS

public:

  explicit
  Node1Xit
  (const std::string entityId,
   Record&           record);

  virtual
  ~Node1Xit();

  // CALLS

public:

  virtual
  const int                                     // returns zero (no duty coupling)
  constrain                                     // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                     // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_nodeCommodity; // 'create' interface calls
  shared_ptr<Cable<C> >     d_cable;
  shared_ptr<Socket<C> >    d_socket;

  // local quantities

  shared_ptr<LmpSoc1>       d_ops;          // specialization required

}; // class 'Node1Xit'

//  ==== XEDOC ================================================
//
//  entity.node-1-xit-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                     > Node1Xit:Elec
//
//        grid spur line node with exit
//
//        my sole normal socket label is 'sock-1'
```

```
//
//      builtin-remark s                        <
//
//      socket-1 l                              > "teas-hv-transmission-1.grid-1"
//
//        socket-1 is my associated bi-directional grid asset
//
//      node-commodity l                        > "cm-electricity-0"
//
//        node-commodity defines the underlying commodity
//
//      nodal-prices [$/J] F                    < 0.0 ..
//
//        to convert to the more usual [$/MWh], multiply the
//        given nodal-prices by 278
//
//  ============================================================

#endif // _NODE01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : node02.h
//  file-create-date : Wed 04-Nov-2009 11:11 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        :  concrete LMP nodes 2 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/node02.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Only some arrangements coded
//
//      This unit covers nodes that, in terms of grid
//      connectivity, support only one incoming and one outgoing
//      grid asset.  This // is clearly a limitation -- although
//      one that would be // straightforward to rectify.

//  HEADER GUARD

#ifndef _NODE02_H_
#define _NODE02_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/node.h"        // LMP node entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

//  CODE

// ----------------------------------------------------------
//  CLASS           : Node2Nul <>
// ----------------------------------------------------------
//  Description  : bidirectional cable, bidirectional socket LMP node
//  Role         : connection between two grid entities
//  Techniques   : class template, 'LmpNul2'
//  Status       : complete
// ----------------------------------------------------------

class LmpNul2;                                // forward (partial) declaration
```

```
template <typename C>                          // 'C' for commodity
class Node2Nul :
  public LmpNode
{
  // DISABLED

private:

  Node2Nul();                                     // zero-argument ctor
  Node2Nul(const Node2Nul& orig);                 // copy constructor
  Node2Nul& operator= (const Node2Nul& orig);     // copy assignment operator

  // CREATORS

public:

  explicit
  Node2Nul
  (const std::string entityId,
   Record&           record);

  virtual
  ~Node2Nul();

  // CALLS

public:

  virtual
  const int                                    // returns zero (no duty coupling)
  constrain                                    // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                    // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_nodeCommodity; // 'create' interface calls
  shared_ptr<Cable<C> >     d_cable;
  shared_ptr<Socket<C> >    d_socket;

  // local quantities

  shared_ptr<LmpNul2>       d_ops;           // specialization required

}; // class 'Node2Nul'

//  ==== XEDOC ================================================
//
//  entity.node-2-nul-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                    > Node2Nul:Elec
//
//        grid node to connect two grid assets
//
//        my sole bi-directional socket label is 'grid-1'
//
//      builtin-remark s                         <
//
//      socket-1 l                               > "teas-transmission-1.grid-1"
//
//        socket-1 is my associated bi-directional grid asset
//
//      node-commodity l                         > "cm-electricity-0"
//
//        node-commodity defines the underlying commodity
//
//      nodal-prices [$/J] F                     < 0.0 ..
//
//        to convert to the more usual [$/MWh], multiply the
//        given nodal-prices by 278
//
```

```
//  ============================================================

// -----------------------------------------------------------
//  CLASS           : Node2Inj <>
// -----------------------------------------------------------
//  Description  : bidirectional cable, bidirectional socket LMP node
//  Role         : connection between two grid entities plus injection
//  Techniques   : class template, 'LmpCab2'
//  Status       : complete
// -----------------------------------------------------------

class LmpCab2;                                  // forward (partial) declaration

template <typename C>                           // 'C' for commodity
class Node2Inj :
  public LmpNode
{
  // DISABLED

private:

  Node2Inj();                                       // zero-argument ctor
  Node2Inj(const Node2Inj& orig);                   // copy constructor
  Node2Inj& operator= (const Node2Inj& orig);       // copy assignment operator

  // CREATORS

public:

  explicit
  Node2Inj
  (const std::string entityId,
   Record&           record);

  virtual
  ~Node2Inj();

  // CALLS

public:

  virtual
  const int                                     // returns zero (no duty coupling)
  constrain                                     // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                     // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_nodeCommodity; // 'create' interface calls
  shared_ptr<Cable<C> >     d_cable1;
  shared_ptr<Cable<C> >     d_cable2;
  shared_ptr<Socket<C> >    d_socket;

  // local quantities

  shared_ptr<LmpCab2>       d_ops;            // specialization required

}; // class 'Node2Inj'

//  ==== XEDOC ================================================
//
//  entity.node-2-inj-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                     > Node2Inj:Elec
//
//        grid node with injection and support for two grid assets
//
//        my sole bi-directional socket label is 'grid-1'
//
//      builtin-remark s                          <
```

```
//
//      socket-1 l                              > "teas-supplier-1.elec-1"
//      socket-2 l                              > "teas-transmission-1.grid-1"
//
//        socket-1 is my normal supplier (injector), socket-2 is
//        my associated bi-directional grid asset
//
//      node-commodity l                        > "cm-electricity-0"
//
//        node-commodity defines the underlying commodity
//
//      nodal-prices [$/J] F                    < 0.0 ..
//
//        to convert to the more usual [$/MWh], multiply the
//        given nodal-prices by 278
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : Node2Xit <>
// ------------------------------------------------------------
//  Description  : bidirectional cable, bidirectional socket LMP node
//  Role         : connection between two grid entities plus exit
//  Techniques   : class template, 'LmpSoc2'
//  Status       : incomplete
// ------------------------------------------------------------

class LmpSoc2;                                  // forward (partial) declaration

template <typename C>                           // 'C' for commodity
class Node2Xit :
  public LmpNode
{
  // DISABLED

private:

  Node2Xit();                                       // zero-argument ctor
  Node2Xit(const Node2Xit& orig);                   // copy constructor
  Node2Xit& operator= (const Node2Xit& orig);       // copy assignment operator

  // CREATORS

public:

  explicit
  Node2Xit
  (const std::string entityId,
   Record&           record);

  virtual
  ~Node2Xit();

  // CALLS

public:

  virtual
  const int                                     // returns zero (no duty coupling)
  constrain                                     // load OSP data
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();                                     // recover OSP results

  // INSTANCE DATA

private:

  // tied quantities

  std::string&              d_nodeCommodity; // 'create' interface calls
  shared_ptr<Cable<C> >     d_cable;
  shared_ptr<Socket<C> >    d_socket1;
  shared_ptr<Socket<C> >    d_socket2;

  // local quantities

  shared_ptr<LmpSoc2>       d_ops;          // specialization required
```

```
}; // class 'Node2Xit'

//  ==== XEDOC ========================================================
//
//  entity.node-2-xit-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//
//      class                                   > Node2Xit:Elec
//
//        grid node with exit and support for two grid assets
//
//        my normal socket label is 'sock-1' and my
//        bi-directional socket label is 'grid-1'
//
//      builtin-remark s                        <
//
//      socket-1 l                              > "teas-transmission-1.grid-1"
//
//        socket-1 is my associated bi-directional grid asset
//
//      node-commodity l                        > "cm-electricity-0"
//
//        node-commodity defines the underlying commodity
//
//      nodal-prices [$/J] F                    < 0.0 ..
//
//        to convert to the more usual [$/MWh], multiply the
//        given nodal-prices by 278
//
//  ===================================================================

#endif // _NODE02_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optctl.h
//  file-create-date : Fri 17-Oct-2008 14:36 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : control OSPs for asset operators / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optctl.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _OPTCTL_H_
#define _OPTCTL_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/optprob.h"     // optimization sub-problem and key sub-classes

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  FORWARD (PARTIAL) DECLARATIONS

class LmpBidSet;

//  CODE

// ----------------------------------------------------------
//  CLASS           : CtlFirstFeasible_A
// ----------------------------------------------------------
//  Description  : implements first feasible solution
//  Role         : supports 'AsopFirstFeasible'
//  Techniques   : supports 'xeona::e_adminFirst'
//  Status       : more-or-less complete -- needs checking
// ----------------------------------------------------------

class CtlFirstFeasible_A :
  public ControlOsp
{
  // DISABLED

private:

  CtlFirstFeasible_A();                                          // zero-argument ctor
  CtlFirstFeasible_A(const CtlFirstFeasible_A& orig);            // copy constructor
  CtlFirstFeasible_A& operator= (const CtlFirstFeasible_A& orig);  // copy assignment oper

  // CREATORS
```

```
public:

  CtlFirstFeasible_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~CtlFirstFeasible_A();

  // FILL PROBLEM CALLS

public:

  void
  uploadNullControl();

  // SPECIALIZED PUSH CALLS

protected:

  virtual
  const int
  pushObj                                   // no value required, simply loads zero
  (const std::string tag = "");

}; // class 'CtlFirstFeasible_A'

// ----------------------------------------------------------
//  CLASS           : CtlMeritOrder_A
// ----------------------------------------------------------
//  Description  : implements prescribed order control
//  Role         : supports 'AsopPrescribedOrder'
//  Techniques   : requires 'xeona::e_adminMerit'
//  Status       : more-or-less complete (except unit limit constraint) -- needs checking
// ----------------------------------------------------------

class CtlMeritOrder_A :
  public ControlOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be local or global cols
  <int>                                     // duty control
  index_type;

  // DISABLED

private:

  CtlMeritOrder_A();                                          // zero-argument constructor
  CtlMeritOrder_A(const CtlMeritOrder_A& orig);               // copy constructor
  CtlMeritOrder_A& operator= (const CtlMeritOrder_A& orig);  // copy assignment operator

  // CREATORS

public:

  CtlMeritOrder_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~CtlMeritOrder_A();

  // FILL PROBLEM CALLS

public:

  index_type                        // duty control gol
  uploadRank
  (const int    rank,               // a merit order from the series { 1, 2, 3, ..., 1024 }
   const double ceilingDuty);       // used to set a "big M"-style structural coefficient

  void
  uploadUnitLimit                   // limit the number of units that can be run at once
  (const int unitLimit);

  // SPECIALIZED PUSH CALLS
```

```
protected:

//    const int
//    pushObj
//    (const double objValue,
//     std::string  tag = "");

private:

  int                d_unitLimit;           // optional limit
  std::vector<int>   d_rankings;            // ascending sorted rank values

}; // class 'CtlMeritOrder_A'

// -----------------------------------------------------------
//  CLASS           : CtlLmpBid_A
// -----------------------------------------------------------
//  Description  : implements LMP (nodal pricing) bidset
//  Role         : supports 'AsopLmpBidStated'
//  Techniques   : requires 'xeona::e_auctionLmp'
//  Status       : more-or-less complete -- needs checking
// -----------------------------------------------------------

class CtlLmpBid_A :
  public ControlOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be local or global cols
  <int>                                     // duty control
  index_type;

  // DISABLED

private:

  CtlLmpBid_A();                                        // zero-argument constructor
  CtlLmpBid_A(const CtlLmpBid_A& orig);                 // copy constructor
  CtlLmpBid_A& operator= (const CtlLmpBid_A& orig);  // copy assignment operator

  // CREATORS

public:

  CtlLmpBid_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~CtlLmpBid_A();

  // FILL PROBLEM CALLS

public:

  index_type                                // duty control gol
  uploadBidSet
  (const shared_ptr<LmpBidSet> bidset);

}; // class 'CtlLmpBid_A'

// -----------------------------------------------------------
//  CLASS           : CtlLeastCost_A
// -----------------------------------------------------------
//  Description  : implements least short-run cost control
//  Role         : supports 'AsopLeastCost'
//  Techniques   : requires 'xeona::e_shortrunModes' -- with cost type supplied by domain
//  Status       : incomplete
//
//  Design notes
//
//      The 'uploadShortrunCosts' call is optional.  However good
//      style might suggest it be called with a zeroed cost set.
//
//      Note the various 'OptimSubProb::downloadShortrunCosts'
//      calls, which also serve our purposed here with
//      modification.
//
// -----------------------------------------------------------
```

```
class CtlLeastCost_A :
  public ControlOsp
{
  // DISABLED

private:

  CtlLeastCost_A();                                          // zero-argument constructor
  CtlLeastCost_A(const CtlLeastCost_A& orig);                // copy constructor
  CtlLeastCost_A& operator= (const CtlLeastCost_A& orig);    // copy assignment operator

  // CREATORS

public:

  CtlLeastCost_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~CtlLeastCost_A();

  // FILL PROBLEM CALLS

public:

  void
  uploadShortrunCosts
  (const CostSet& shiftCosts);

}; // class 'CtlLeastCost_A'

// ----------------------------------------------------------
//  CLASS           : CtlQuan_A
// ----------------------------------------------------------
//  Description  : implements demand quantity
//  Role         : supports 'AsopInelasticTs'
//  Techniques   : (nothing special)
//  Status       : more-or-less complete -- needs checking
// ----------------------------------------------------------

class CtlQuan_A :
  public ControlOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                       // can be local or global cols
  <int>                                      // duty control
  index_type;

  // DISABLED

private:

  CtlQuan_A();                                    // zero-argument constructor
  CtlQuan_A(const CtlQuan_A& orig);               // copy constructor
  CtlQuan_A& operator= (const CtlQuan_A& orig);   // copy assignment operator

  // CREATORS

public:

  CtlQuan_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~CtlQuan_A();

  // FILL PROBLEM CALLS

public:

  index_type                                 // duty control gol
  uploadControl
  (const double demand);                     // demanded quantity

  // INSTANCE DATA
```

```
private:

  index_type    d_cols;                     // local cols

}; // class 'CtlQuan_A'

#endif // _OPTCTL_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optgate.h
//  file-create-date : Wed 25-Mar-2009 21:08 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : various OSPs for gateways / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optgate.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _OPTGATE_H_
#define _OPTGATE_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/optprob.h"     // optimization sub-problem and key sub-classes

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  FORWARD (PARTIAL) DECLARATIONS

class BandedTariffSet;                      // an 'uploadTariffSet' argument

//  CODE

// ------------------------------------------------------------
//  CLASS           : QanTechCapacity_A
// ------------------------------------------------------------
//  Description  : create a technical capacity upper bound OSP
//  Role         : concrete gateways
//  Techniques   : (nothing special)
//  Status       : incomplete
// ------------------------------------------------------------

class QanTechCapacity_A :
  public QuantityOsp
{
  // USING DECLARATIONS

protected:

  using OptimSubProb::s_logger;             // place in common scope for this class

  // TYPEDEFS

public:
```

```
  typedef boost::tuple                      // can be in either local or global cols
  <int>                                     // coupling index for internal use
  index_type;

  typedef boost::tuple
  <double>                                  // technical duty (recovered)
  results_type;

  // CREATORS

public:

  QanTechCapacity_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string          tag);

  virtual
  ~QanTechCapacity_A();

  // FILL PROBLEM CALLS

public:

  index_type
  uploadCapacity                            // wrapper to two argument version
  (const std::pair<double, double> capacity);

  index_type
  uploadCapacity
  (const double loCapacity,
   const double hiCapacity);

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols

}; // class 'QanTechCapacity_A'

// ------------------------------------------------------------
//  CLASS           : OfrTariffSet_A
// ------------------------------------------------------------
//  Description  : create a tariff set offer OSP
//  Role         : concrete gateways
//  Techniques   : (nothing special)
//  Status       : incomplete
// ------------------------------------------------------------

class OfrTariffSet_A :
  public OfferOsp
{
  // USING DECLARATIONS

protected:

  using OptimSubProb::s_logger;             // place in common scope for this class

  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int>                                     // coupling index for internal use
  index_type;

  typedef boost::tuple
  <double,                                  // marginal price ($/quantity)
   double,                                  // fixed component ($)
   double,                                  // total price ($)
   double>                                  // transaction (quantity)
  results_type;

private:
```

```
  typedef std::pair
  <double,                                  // band
   double>                                  // price
  tariff_type;

  // DISABLED

private:

  OfrTariffSet_A();                                          // zero-argument constructor
  OfrTariffSet_A(const OfrTariffSet_A& orig);                // copy constructor
  OfrTariffSet_A& operator= (const OfrTariffSet_A& orig);    // copy assignment operator

  // CREATORS

public:

  OfrTariffSet_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string          tag);

  // UPLOAD CALLS

public:

  index_type
  uploadTariffSet
  (const shared_ptr<BandedTariffSet> tariffset,
   const double                      capacity);

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution
  (const int interval) const;               // horizon interval

  // INSTANCE DATA

private:

  index_type                    d_cols;          // local cols
  shared_ptr<BandedTariffSet>   d_tariffset;     // stored so solution can be interpreted
  double                        d_capacity;      // ditto

}; // class 'OfrTariffSet_A'

// ----------------------------------------------------------
//  CLASS           : QanObligToSupply_A
// ----------------------------------------------------------
//  Description  : create obligation to suppy as an equality constraint OSP
//  Role         : concrete gateways
//  Techniques   : (nothing special)
//  Status       : incomplete
// ----------------------------------------------------------

class QanObligToSupply_A :
  public QuantityOsp
{
  // USING DECLARATIONS

protected:

  using OptimSubProb::s_logger;             // place in common scope for this class

  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int>                                     // coupling index for internal use
  index_type;

  typedef boost::tuple
  <double>                                  // obligation (recovered)
  results_type;

  // DISABLED

private:
```

```
  QanObligToSupply_A();                                          // zero-argument ctor
  QanObligToSupply_A(const QanObligToSupply_A& orig);            // copy constructor
  QanObligToSupply_A& operator= (const QanObligToSupply_A& orig); // copy assignment op

  // CREATORS

public:

  QanObligToSupply_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string          tag);

  // UPLOAD CALLS

  index_type
  uploadObligation
  (const double supplyObligation);

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                      // local cols

}; // class 'QanObligToSupply_A'

#endif // _OPTGATE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optjunc.h
//  file-create-date : Mon 26-Oct-2009 10:26 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : operations OSPs for junctions / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optjunc.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _OPTJUNC_H_
#define _OPTJUNC_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/optprob.h"     // optimization sub-problem (OSP) and key sub-classes

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ------------------------------------------------------------
//  CLASS           : JncSplit2
// ------------------------------------------------------------
//  Description  : demand splitter lacking exergy destruction or cost formation
//  Role         : for instance, 'JuncDemand2Split'
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

class JncSplit2 :
  public JunctionOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // socket (single)
   int,                                     // cable-1
   int>                                     // cable-2
  index_type;

  typedef boost::tuple
  <double>                                  // socket output
  results_type;
```

```
  // CREATORS

public:

  JncSplit2
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~JncSplit2();

  // FILL PROBLEM CALLS
public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols

}; // class 'JncSplit2'

// ----------------------------------------------------------
//  CLASS           : JncSplit3
// ----------------------------------------------------------
//  Description  : demand splitter lacking exergy destruction or cost formation
//  Role         : for instance, 'JuncDemand3Split'
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class JncSplit3 :
  public JunctionOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // socket (single)
   int,                                     // cable-1
   int,                                     // cable-2
   int>                                     // cable-3
  index_type;

  typedef boost::tuple
  <double>                                  // socket output
  results_type;

  // CREATORS

public:

  JncSplit3
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~JncSplit3();

  // FILL PROBLEM CALLS
public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA
```

```
private:

  index_type    d_cols;                       // local cols

}; // class 'JncSplit3'

// ----------------------------------------------------------
//  CLASS           : JncJoin2
// ----------------------------------------------------------
//  Description  : demand joiner lacking exergy destruction or cost formation
//  Role         : for instance, 'JuncDemand2SJoin'
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class JncJoin2 :
  public JunctionOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                       // can be in either local or global cols
  <int,                                      // cable (single)
   int,                                      // socket-1
   int>                                      // socket-2
  index_type;

  typedef boost::tuple
  <double>                                   // cable input
  results_type;

  // CREATORS

public:

  JncJoin2
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~JncJoin2();

  // FILL PROBLEM CALLS
public:

  index_type                                 // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                      // local cols

}; // class 'JncJoin2'

#endif // _OPTJUNC_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optnode.h
//  file-create-date : Tue 03-Nov-2009 19:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : node optimization sub-problems for LMP nodes / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optnode.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _OPTNODE_H_
#define _OPTNODE_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/optprob.h"     // optimization sub-problem (OSP) class and key sub-classes

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ------------------------------------------------------------
//  CLASS           : LmpCab1
// ------------------------------------------------------------
//  Description  : LMP node with normal cable and one bidirectional connection
//  Role         : for instance, 'Node1Inj' (spur line injection node)
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

class LmpCab1 :
  public LmpNodeOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // cable     (single, normal)
   int>                                     // socket-bi (single, bidirectional)
  index_type;

  typedef boost::tuple
  <double,                                  // injection quantity
   double>                                  // nodal price (reduced cost)
```

```
  results_type;

  // CREATORS

public:

  LmpCab1
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~LmpCab1();

  // FILL PROBLEM CALLS

public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols
  int           d_row;                      // local row

}; // class 'LmpCab1'

// ------------------------------------------------------------
//  CLASS           : LmpSoc1
// ------------------------------------------------------------
//  Description  : LMP node with normal socket and one bidirectional connection
//  Role         : for instance, 'Node1Xit' (spur line exit node)
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

class LmpSoc1 :
  public LmpNodeOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // cable-bi (single, bidirectional)
   int>                                     // socket   (single, normal)
  index_type;

  typedef boost::tuple
  <double,                                  // exit quantity
   double>                                  // nodal price (reduced cost)
  results_type;

  // CREATORS

public:

  LmpSoc1
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~LmpSoc1();

  // FILL PROBLEM CALLS

public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
```

```
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols
  int           d_row;                      // local row

}; // class 'LmpSoc1'

// ----------------------------------------------------------
//  CLASS           : LmpNul2
// ----------------------------------------------------------
//  Description  : LMP node two bidirectional connections
//  Role         : for instance, 'Node2Nul' (node)
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class LmpNul2 :
  public LmpNodeOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // cable-bi  (single, bidirectional)
   int>                                     // socket-bi (single, bidirectional)
  index_type;

  typedef boost::tuple
  <double,                                  // throughput
   double>                                  // nodal price (reduced cost)
  results_type;

  // CREATORS

public:

  LmpNul2
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~LmpNul2();

  // FILL PROBLEM CALLS

public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols
  int           d_row;                      // local row

}; // class 'LmpNul2'

// ----------------------------------------------------------
//  CLASS           : LmpCab2
// ----------------------------------------------------------
//  Description  : LMP node with normal cable and two bidirectional connections
//  Role         : for instance, 'Node2Inj' (node)
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class LmpCab2 :
  public LmpNodeOsp
{
```

```
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // cable-1    (normal)
   int,                                     // cable-2-bi (bidirectional)
   int>                                     // socket-bi  (single, bidirectional)
  index_type;

  typedef boost::tuple
  <double,                                  // throughput
   double>                                  // nodal price (reduced cost)
  results_type;

  // CREATORS

public:

  LmpCab2
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~LmpCab2();

  // FILL PROBLEM CALLS

public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols
  int           d_row;                      // local row

}; // class 'LmpCab2'

// ----------------------------------------------------------
//  CLASS           : LmpSoc2
// ----------------------------------------------------------
//  Description  : LMP node with normal socket and two bidirectional connections
//  Role         : for instance, 'Node2Xit' (node)
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class LmpSoc2 :
  public LmpNodeOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // cable-bi    (bidirectional)
   int,                                     // socket-1    (normal)
   int>                                     // socket-2-bi (single, bidirectional)
  index_type;

  typedef boost::tuple
  <double,                                  // exit quantity
   double>                                  // nodal price (reduced cost)
  results_type;

  // CREATORS

public:

  LmpSoc2
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);
```

```
  virtual
  ~LmpSoc2();

  // FILL PROBLEM CALLS

public:

  index_type                                // global cols
  uploadEngineering();

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols
  int           d_row;                      // local row

}; // class 'LmpSoc2'

#endif // _OPTNODE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optops.h
//  file-create-date : Fri 17-Oct-2008 14:40 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : operations OSPs for technical assets / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optops.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  OSP class naming conventions
//
//      Take class "OpsDevice_A" for instance -- this refers to
//      implementation A of the OpsDevice optimization
//      sub-problem.
//
//      A second implementation B would retain the same interface
//      -- that is, all public function declarations would remain
//      identical -- and only the internals would change.
//
//      Moreover, there should be no requirement that
//      implementation B should mimic the behavior of
//      implementation A.  But the two behaviors should be
//      documented and preferably by difference if possible.

//  HEADER GUARD

#ifndef _OPTOPS_H_
#define _OPTOPS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/optprob.h"     // optimization sub-problem and key sub-classes

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  CODE

// ----------------------------------------------------------
//  CLASS           : OpsTransmission_A
// ----------------------------------------------------------
//  Description  : discretized two-way transmission line
//  Role         : 'TeasHvTransmission'
//  Techniques   : discretization
//  Status       : first-pass complete but needs careful checking
// ----------------------------------------------------------

class OpsTransmission_A :
  public OperationsOsp
```

```
{
  // TYPEDEFS

public:

  typedef boost::tuple                      // can be in either local or global cols
  <int,                                     // cable (single, bidirectional)
   int>                                     // socket (single, bidirectional)
  index_type;

  typedef boost::tuple
  <bool,                                    // flow direction flag, 'true' = forward
   double,                                  // cable flow [W]
   double,                                  // socket flow [W]
   double,                                  // relative duty [-]
   double>                                  // relative loss [-]
  results_type;

  // CREATORS

public:

  OpsTransmission_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~OpsTransmission_A();

  // FILL PROBLEM CALLS

public:

  // lower level processing (principal problem building call)
  index_type                                // global cols
  uploadEngineering
  (const double injectCapacity,             // in unprefixed SI units [W]
   const double voltage,                    // in unprefixed SI units [V]
   const double ohmsPerMetre,               // in unprefixed SI units [ohm/m]
   const double length,                     // in unprefixed SI units [m]
   const int    discretization);            // number of discretization steps

  // higher level call (convenience call), see the implementation
  // file for more information
  index_type                                // global cols
  uploadEngineering
  (const double injectCapacity,             // injection capacity
   const double maxRelLosses,               // relative losses at capacity (not percent)
   const int    discretization);            // number of discretization steps

  // DOWNLOAD SOLUTION CALLS

public:

  results_type                              // see local typedef
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols
  double        d_injectCapacity;

}; // class 'OpsTransmission_A'

// ----------------------------------------------------------
//  CLASS           : OpsFac1Out1_A
// ----------------------------------------------------------
//  Description  : two-port conversion with constant marginal efficiency and shutdown mode
//  Role         : for instance, 'TeasOxidToElec'
//  Techniques   : binary variables
//  Status       : first-pass complete but needs careful checking
// ----------------------------------------------------------

class OpsFac1Out1_A :
  public OperationsOsp
{
  // TYPEDEFS

public:
```

```
  typedef boost::tuple                          // can be in either local or global cols
  <int,                                         // fuel stream
   int,                                         // product stream (typically for coupling)
   int>                                         // trip status (nonsensical for coupling)
  index_type;

  typedef boost::tuple
  <double,                                      // fuel usage (factor) [W]
   double,                                      // product (output) [W]
   bool>                                        // trip status, 'true' = ran
  results_type;

  // CREATORS

public:

  OpsFac1Out1_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~OpsFac1Out1_A();

  // FILL PROBLEM CALLS

public:

  index_type                                    // global cols
  uploadEngineering                             // J-to-J form
  (const double prodLoBound,                    // lower output below which the asset trips
   const double prodHiBound,                    // upper output
   const double marginalEfficiency,             // slope (as decimal not percentage)
   const double fuelNoload,                     // fuel use on idle (input-axis intercept)
   const double fuelAncillary,                  // fuel use on shutdown (for ancillaries)
   const double rampLoSize,                     // ramp down restriction
   const double rampHiSize);                    // ramp up restiction

  index_type                                    // global cols
  uploadEngineering                             // kg-to-J form
  (const double prodLoBound,                    // lower output below which the asset trips
   const double prodHiBound,                    // upper output
   const double marginalEfficiency,             // slope (as decimal not percentage)
   const double fuelNoload,                     // fuel use on idle (input-axis intercept)
   const double fuelAncillary,                  // fuel use on shutdown (for ancillaries)
   const double rampLoSize,                     // ramp down restriction
   const double rampHiSize,                     // ramp up restriction
   const double specEnthalpy);                  // kg->J conversion

  // DOWNLOAD SOLUTION CALLS

  results_type
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                         // local cols

}; // class 'OpsFac1Out1'

// ------------------------------------------------------------
//  CLASS           : OpsFac0Out1_A
// ------------------------------------------------------------
//  Description  : harvest/extract/import source-style OSP
//  Role         : for instance, 'TeasWindfarm'
//  Techniques   : low (often zero) and high bounds
//  Status       : first-pass complete but needs careful checking
//
//  Design notes
//
//      The concept of a "source" entity covers harvest,
//      extraction, and un-modeled import.
//
//      The concept also imply a "reservoir" in the thermodynamic
//      and economic senses -- that is, an invariant intensive
//      state is assumed, irrespective of rate of take.
//
// ------------------------------------------------------------
```

```
class OpsFac0Out1_A :
  public OperationsOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                        // can be local or global cols
  <int>                                       // output variable
  index_type;

  typedef boost::tuple
  <double>                                    // actual output [W]
  results_type;

  // CREATORS

public:

  OpsFac0Out1_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~OpsFac0Out1_A();

  // FILL PROBLEM CALLS

public:

  index_type                                  // global cols
  uploadEngineering
  (const double prodLoBound,                  // lower output (typically zero)
   const double prodHiBound);                 // upper output

  // DOWNLOAD SOLUTION CALLS

  results_type                                // see local typedef
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                       // local cols

}; // class 'OpsFac0Out1'

// ------------------------------------------------------------
//  CLASS           : OpsFac1Out0_A
// ------------------------------------------------------------
//  Description  : load/export sink-style OSP
//  Role         : for instance, 'TeasLoadElecTs'
//  Techniques   : low (often zero) and high bounds, single bound
//  Status       : first-pass complete but needs careful checking
//
//  Design notes
//
//      The concept of a "isnk" entity covers load and un-modeled
//      export.
//
//      The concept also imply a "reservoir" in the thermodynamic
//      and economic senses -- that is, an invariant intensive
//      state is assumed, irrespective of rate of disposal.
//
// ------------------------------------------------------------

class OpsFac1Out0_A :
  public OperationsOsp
{
  // TYPEDEFS

public:

  typedef boost::tuple                        // can be local or global cols
  <int>                                       // output variable
  index_type;

  typedef boost::tuple
  <double>                                    // actual input [W]
  results_type;
```

```
  // CREATORS

public:

  OpsFac1Out0_A
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode);

  virtual
  ~OpsFac1Out0_A();

  // FILL PROBLEM CALLS

public:

  index_type                                // global cols
  uploadEngineering
  (const double takeFixed);                 // fixed input (can be zero)

  index_type                                // global cols
  uploadEngineering
  (const double takeLoBound,                // lower input (typically zero)
   const double takeHiBound);               // upper input

  // DOWNLOAD SOLUTION CALLS

  results_type                              // see local typedef
  downloadSolution() const;

  // INSTANCE DATA

private:

  index_type    d_cols;                     // local cols

}; // class 'OpsFac1Out0'

#endif // _OPTOPS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optprob.h
//  file-create-date : Fri 17-Oct-2008 08:19 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : optimization sub-problem (OSP) and key sub-classes / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optprob.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _OPTPROB_H_
#define _OPTPROB_H_

//  OVERVIEW
//
//  Design strategy
//
//      OSP means "optimization sub-problem" and refers to all
//      concrete classes derived from the 'OptimSubProb' abstract
//      base class.
//
//      On first view, the introduction of an OSP class hierarchy
//      may appear an unnecessary complication -- after all, the
//      equivalent code could be buried in the entities
//      themselves (whilst noting that that approach is not
//      precluded).  However the selected design should reduce
//      code repetition and duplicated testing.
//
//      The scheme given here therefore tries to provide a set of
//      relatively generic OSPs for use by entity authors with
//      the aim of ultimately reducing the associated coding and
//      commissioning overhead.
//
//  CAUTION: problem build requires EXCLUSIVE access
//
//      This code assumes each 'OptimSubProb' derived instance
//      has exclusive access to the relevant solver interface
//      'svif::SolverIf' object during problem loading.
//      Therefore DO NOT INTERLACE problem loading code.
//
//      Enforcement via build locking, for instance, is not
//      provided (but perhaps should be).
//
//      Further details are given in the implementation file.
//
//  Terminology and usage
//
//      The following naming convention for OSP member functions
//      is employed:
//
```

```
//          internal (protected) calls begin : "push" "repush" "recover"
//          external (public) calls begin    : "upload" "download"
//
//      The following indexing definitions and terminology are used:
//
//          columns represent variables (strictly speaking, structural variables)
//          rows represent constraints (strictly speaking, axillary variables)
//
//          col : local col using the 'd_colStart' offset
//          row : local row using the 'd_rowStart' offset
//          gol : global col
//          gow : global row
//
//      If need be, calls regarding variables should be named
//      thus:
//
//          getXxxx
//          setXxxx    also returns previous value
//          Xxxx       reference (alias) variant uses identical name
//
//      Problem building calls return:
//
//          void
//          boost::tuple<int>
//          boost::tuple<int, int, ..>
//
//      Problem building calls more specifically:
//
//          uploadEngineering      technical description
//          uploadShortrunCosts    duty and factor specific costs,
//                                   overwrite or increment "shift" *
//          uploadBidset           LMP (nodal pricing) bid set
//          uploadCeilingDuty      current upper capacity = maximum duty
//          uploadFloorDuty        current lower capacity = must-run duty
//
//          * overloaded function, meaning depends on signature
//
//      Solution recovery calls:
//
//          boost::tuple<CostSet> results = downloadActivityCost
//          results_type          results = downloadSolution
//
//  One and zero-based indexing
//
//      The rows and cols (and thereby gows and gols) utilize
//      ONE-based indexing.  This aligns with the conventions
//      employed by GLPK.
//
//      The various objective coefficient vectors use ZERO-based
//      indexing -- the 0th entry is for the so-called "shift"
//      constant term.
//
//      Boost.Tuple tuples use ZERO-based indexing.
//
//      For completeness, the step count is also ZERO-based.
//
//      In general, pay attention to the indexing base and look
//      for off-by-one errors!
//
//  Duty not for OSPs
//
//       OSPs do not support the notion of "duty" although they
//       must and do return the gols to their various input and
//       output streams.  Rather, duty and duty coupling are
//       concepts for the technical asset and/or asset operator
//       and, more particularly, must be resolved within their
//       'constrain' calls.
//
//  Tuple usage
//
//       Function arguments and function returns that could
//       potentially utilize more than one value are ALWAYS
//       declared as Boost.Tuple tuples.  This means, somewhat
//       awkwardly perhaps, the use of 1-tuple arguments and
//       1-tuple returns where only one value is required.  The
//       upside is that a consistent idiom can be maintained and
//       that the notion of multiple dimensionality is reinforced.
//
//  Scheme (may not be current)
//
//       OSP derivative        object prefix   unit *      typical host
```

```
//      ---------------------------------------------------------------
//      CouplingOsp           **              optprob     couple ***
//      ConnectionOsp         con             optprob     Interface ****
//      sub-OperationsOsp     ops             optops      TechnicalAsset
//                                            optgate     Gateway
//                                            optjunc     DomainJunction
//                                            optnode     HvNode
//      sub-ControlOsp        ctl             optctl      AssetOperator
//      sub-QuantityOsp       qan             optgate     Gateway
//      sub-Offer             ofr             optgate     Gateway
//      ---------------------------------------------------------------
//      "sub-" indicates a derived class hierarchy
//        * = translation unit (read "header") which holds concrete specializations
//       ** = use free function 'xeona::couple' in preference to dedicated ctor call
//      *** = unlike other examples, OSP only last for duration of call
//     **** = a grep search reveals that the  "con"  is probably not used
//
//  CLASS HIERARCHY (ongoing and possibly out-of-date)
//
//                                            unit
//      ---------------------------------------------------------------
//      - OptimSubProb                      -+
//          + CouplingOsp                    |   optprob (here)
//          + ConnectionOsp                 -+
//          - OperationsOsp                      optops
//              + OpsXxxx_X
//                ...
//          - ControlOsp                         optctl
//              + CtlXxxx_X
//                ...
//          - QuantityOsp                   -+
//              + QanTechCapacity_X          |
//              + QanObligToSupply_X         |   optgate
//          - OfferOsp                       |
//              + OfrTariffSet_X            -+
//          - JunctionOsp
//              + JncSplit2                      optjunc
//                ...
//          - LmpNodeOsp
//              + LmpCab1                        optnode
//                ...
//      ---------------------------------------------------------------
//        - = abstract  + = concrete

//  LOCAL AND SYSTEM INCLUDES

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
#include "../c/costs.h"       // cost sets and support

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>        // n-tuples, ref(), cref()

//  FORWARD (PARTIAL) DECLARATIONS

class BandedTariffSet;                      // an 'uploadTariffSet' argument

//  CODE

// ---------------------------------------------------------
//  CLASS           : OptimSubProb (abstract)
// ---------------------------------------------------------
//  Description  : entity-local abstraction to the domain-specific solver interface
//  Role         : abstract base class for all '*Osp' classes
//  Techniques   : "push loading" (see below), sub-classes regularly use 'boost::tuple'
//  Status       : complete
// ---------------------------------------------------------

class OptimSubProb
{
  // DISABLED

private:

  OptimSubProb(const OptimSubProb& orig);              // copy constructor
```

```
  OptimSubProb& operator= (const OptimSubProb& orig);  // copy assignment operator

  // CREATORS

public:

  OptimSubProb();                              // zero-argument constructor

  // CAUTION: zero-argument constructor should only be used by
  // the base-class 'Interface' constructor

  OptimSubProb
  (shared_ptr<svif::SolverIf> solver,        // solver interface object
   const xeona::DomainMode    commitmentMode,
   const std::string&         ospDesc = ""); // hardcode for logs, see also 'loadOspLabel'

  virtual
  ~OptimSubProb() = 0;                       // create abstract class

  // HOST-RELATED CALLS - used by assets and operators

public:

  std::string
  loadOspLabel                               // optional OSP labeling
  (const std::string label);                 // sub-problem (and not solver) label

  //  CALLS PROVIDED FOR UNIT TEST PURPOSES

public:

  bool                                       // should always be 'true'
  reportBuildIntegrity                       // can call any time
  (const std::string comment = "");

  // PUSH CALLS (and similar)

protected:

  // virtual

  virtual
  void
  pushIncShift                               // increment objective function constant term
  (const double      shiftValue,             // non-short-run commitment mode
   const CostSet&    specCosts);

  virtual
  void
  pushIncShift                               // increment objective function constant term
  (const CostSet&    shiftValues);           // all commitment modes

  virtual
  const int                                  // the col index just employed
  pushObj
  (const double      objValue,               // for non-short-run commitment
   const CostSet&    specCosts,
   const std::string tag = "");

  virtual
  const int                                  // the col index just employed
  pushObj
  (const CostSet&    specCosts,
   const std::string tag = "");

  virtual
  const double                               // return the previous value, now overwritten
  repushObj
  (const int         col,
   const double      objValue,
   const CostSet&    specCosts);

  virtual
  const double                               // return the previous value, now overwritten
  repushObj
  (const int         col,
   const CostSet&    specCosts);

  // non-virtual

  const int                                  // the same col index for convenience
```

```
markBinary                                  // in set { 0, 1 }
(const int col);

const int                                   // the same col index for convenience
markInteger
(const int col);

const int                                   // the row index just employed
pushRhs
(const double                rhsValue,
 const svif::ConstraintSense conSense,      // { svif::L, E, G, R, N } from unit 'siglp'
 const std::string           tag = "");     // passed thru to solver

const int                                   // number of local nonzero structural coeffs
pushCof
(const int    row,
 const int    col,
 const double coeffValue);

const int                                   // always zero, 'd_cofCount' not incremented
pushGof                                     // like 'pushCof' but with global row and col
(const int    gow,                          // global row
 const int    gol,                          // global col
 const double coeffValue);

int                                         // number of bound changes {0, 1, 2}
openBnds                                    // set to [-inf, +inf]
(const int    col);

int                                         // number of bound changes {0, 1, 2}
changeBnds                                  // CAUTION: not expected to be used
(const int    col,
 const double lowerBnd,
 const double upperBnd);

void
resetBnds                                   // reset to defaults, see unit 'd/siglpk'
(const int    col);

// CALLS WHICH TRANSLATE OSP INDEXES

// int variants

const int
globalcol                                   // add 'd_colStart'
(const int localCol) const;

const int
globalrow                                   // add 'd_rowStart'
(const int localRow) const;

const int
localcol                                    // subtract 'd_colStart'
(const int globalCol) const;

const int
localrow                                    // subtract 'd_rowStart'
(const int globalRow) const;

// tuple variants covering one to nine indexes

boost::tuple<int>
globalcols(boost::tuple<int>) const;

boost::tuple<int, int>
globalcols(boost::tuple<int, int>) const;

boost::tuple<int, int, int>
globalcols(boost::tuple<int, int, int>) const;

boost::tuple<int, int, int, int>
globalcols(boost::tuple<int, int, int, int>) const;

boost::tuple<int, int, int, int, int>
globalcols(boost::tuple<int, int, int, int, int>) const;

boost::tuple<int, int, int, int, int, int>
globalcols(boost::tuple<int, int, int, int, int, int>) const;

boost::tuple<int, int, int, int, int, int, int>
globalcols(boost::tuple<int, int, int, int, int, int, int>) const;
```

```
  boost::tuple<int, int, int, int, int, int, int, int>
  globalcols(boost::tuple<int, int, int, int, int, int, int, int>) const;

  boost::tuple<int, int, int, int, int, int, int, int, int>
  globalcols(boost::tuple<int, int, int, int, int, int, int, int, int>) const;

  // PROBLEM GET CALLS - can be made anytime but returned value
  // may not be complete

  CostSet
  getShift() const;                           // current objective function "shift" term

  std::vector<CostSet>
  getObjs() const;

  // SOLUTION RECOVERY CALLS - note the individual and vector versions

public:

  // each recovery call tests "d_solver->isUsableSoln() == true"
  // and will complain and react accordingly

  double
  downloadVar                                 // col primal value for underlying LP
  (const int col) const;

  std::vector<double>
  downloadVars() const;

  const bool                                  // 'true' means successful calculation
  downloadVarCosts                            // excludes "shift" term
  (CostSet& accruedCosts) const;

  const bool                                  // 'true' means successful calculation
  downloadShortrunCosts                       // pass-by-reference form
  (CostSet& varCosts,                         // identical to 'downloadVarCosts'
   CostSet& fixCosts) const;                  // assign "shift" term

  boost::tuple
  <CostSet,                                   // "var" costs
   CostSet>                                   // "fix" costs
  downloadShortrunCosts() const;              // tuple form

  double
  downloadSlack                               // row dual value for underlying LP
  (const int row) const;

  std::vector<double>
  downloadSlacks() const;                     // slack values

  // UTILITY FUNCTIONS

protected:

  double
  getInf() const
  {
    return std::numeric_limits<double>::infinity();    // refer <limits>
  }

  double
  getNaN() const
  {
    return std::numeric_limits<double>::quiet_NaN();   // refer <limits>
  }

  // INSTANCE AND STATIC RESET CALLS
  // PONDER: which of these reset calls are needed?

protected:

  void
  resetSolver                                 // will affect all other solver users
  (const std::string label = "");             // solver (and not sub-problem) label

  static
  void
  resetSolver                                 // will affect all other solver users
  (shared_ptr<svif::SolverIf> solver,
   const std::string          label = "");   // solver (and not sub-problem) label
```

```
  // INSTANCE DATA

protected:

  std::string           d_label;           // optional OSP label (for tags)
  const std::string     d_ospDesc;         // optional OSP description (for logs)

protected:

  // CAUTION: indexing: the sub-problem uses one-based indexing
  // for rows and cols while the various objective coefficient
  // vectors use zero-based indexing -- code utilizing the
  // objective vectors is not common, but do pay attention for
  // off-by-one errors!

  const int             d_rowStart;        // global and acquired from solver
  const int             d_colStart;

  int                   d_rowCount;        // local, initially zero
  int                   d_colCount;
  int                   d_cofCount;        // local nonzero structural coefficients

  int                   d_colTrack;        // local, ad-hoc index store, initially zero
  int                   d_rowTrack;

  // these containers are not necessarily used by all the various
  // sub-classes but they are all placed in the base class -- if
  // need be, some can be moved down the inheritance hierarchy
  // (although there is no compelling reason to do so)

  std::vector<double>   d_finObjs;         // also specific costs, note zero-based index
  std::vector<double>   d_ghgObjs;
  std::vector<double>   d_noxObjs;
  std::vector<double>   d_depObjs;
  std::vector<double>   d_lucObjs;

  std::vector<double>   d_bidObjs;         // bids
  std::vector<double>   d_mitObjs;         // merit order
  std::vector<double>   d_nulObjs;         // first feasible (always zero)

  // CAUTION: 'd_commitmentMode' and 'd_solver' should only be
  // used by protected 'push' calls and not by public
  // 'uploadProblem' calls -- unfortunately this requirement is
  // not enforced

  const xeona::DomainMode       d_commitmentMode;      // implies a cost type too
  shared_ptr<svif::SolverIf>    d_solver;              // interface to GLPK solver

  // STATIC DATA

protected:

  static logga::spLogger    s_logger;        // shared_ptr to single logger object

}; // class 'OptimSubProb'

// ------------------------------------------------------------
//  CLASS           : CouplingOsp
// ------------------------------------------------------------
//  Description  : concrete class for "coupling" two OSPs
//  Role         : used in 'constrain' calls from 'AssetOperator' instances and similar
//  Techniques   : concrete
//  Status       : complete
//  See also     : free function 'couple' (below)
// ------------------------------------------------------------

class CouplingOsp :
  public OptimSubProb
{
  // CREATORS

public:

  CouplingOsp
  (shared_ptr<svif::SolverIf> solver);

  virtual
  ~CouplingOsp();

  // PUBLIC FUNCTIONS
```

```
public:

  bool                                      // 'true' if coupling constraint added
  coupleGols
  (const int         golA,
   const int         golB,
   const std::string tag = "");

}; // class 'CouplingOsp'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::couple
// ----------------------------------------------------------
//  Description  : "couple" two OSPs conveniently
//  Role         : used in 'constrain' calls from 'AssetOperator' instances and similar
//  Techniques   : free function wrapper to 'Coupling::coupleGols'
//  Status       : complete
// ----------------------------------------------------------

namespace xeona
{
  bool                                      // 'true' if coupling constraint added
  couple
  (shared_ptr<svif::SolverIf> solver,
   const int                  golA,         // usually from { ops }
   const int                  golB,         // usually from { ctl connection }
   const std::string          tag = "");

} // namespace 'xeona'

// ----------------------------------------------------------
//  CLASS           : ConnectionOsp
// ----------------------------------------------------------
//  Description  : concrete class for "connecting" block interfaces in an OSP context
//  Role         : used in 'constrain' calls from 'TechnicalAsset' instances and similar
//  Techniques   : concrete
//  Status       : complete
// ----------------------------------------------------------

class ConnectionOsp :
  public OptimSubProb
{
  // CREATORS

public:

  ConnectionOsp();                          // zero-argument constructor

  ConnectionOsp
  (shared_ptr<svif::SolverIf> solver,
   const int                  passCount = 0);

  virtual
  ~ConnectionOsp();

  // PUBLIC FUNCTIONS

public:

  int                                       // revised value
  incPassCount                              // increment pass count, starting zero
  (shared_ptr<svif::SolverIf> solver);

  int                                       // current value
  getPassCount() const;

  void
  resetPassCount();                         // reset to zero

  void
  storeGol
  (const int gol);

  int
  recoverGol() const;

  bool                                      // 'true' if connection constraint added
  bindGols
  (const int         gol1,
   const int         gol2,
```

```
   const std::string tag = "");

  // UTILITY FUNCTIONS

private:

  bool                                      // 'false' if problem detected
  checkSolverConsistency                    // used by 'incPassCount'
  (const int                  passCount,
   shared_ptr<svif::SolverIf> solver,
   std::string&               info);        // processing information for local use

  // INSTANCE DATA

private:

  int            d_gol;                     // stored global col
  int            d_passCount;               // pass count in { 0, 1, 2 }
  std::string    d_lastSolverAddress;       // determined by stream insertion

}; // class 'ConnectionOsp'

// ----------------------------------------------------------
//  CLASS           : ControlOsp (abstract)
// ----------------------------------------------------------
//  Description  : abstract class covering penalty-based control (PBC)
//  Role         : parent for concrete "Ctl" sub-classes
//  Techniques   : pure virtual destructor
//  Status       : complete
// ----------------------------------------------------------

class ControlOsp :
  public OptimSubProb
{
  // CREATORS

public:

  ControlOsp
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const xeona::DomainMode    commitmentModeSum,
   const std::string&         ospDesc = "(not overwritten)");

  virtual
  ~ControlOsp() = 0;                        // create abstract class

  // FILL PROBLEM CALLS

public:

  void
  setFloorDuty
  (const double mustRunDuty,
   const int    teasDutyCol);

  void
  setCeilingDuty
  (const double maxDuty,
   const int    teasDutyCol);

  // INSTANCE DATA

protected:

  int    d_dutyCol;                         // local col

}; // class 'ControlOsp'

// ----------------------------------------------------------
//  CLASS           : OperationsOsp (abstract)
// ----------------------------------------------------------
//  Description  : abstract class covering operations
//  Role         : parent for concrete "Ops" sub-classes
//  Techniques   : pure virtual destructor
//  Status       : complete
// ----------------------------------------------------------

class OperationsOsp :
  public OptimSubProb
{
```

```
  // CREATORS

public:

  OperationsOsp
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string&         ospDesc = "(not overwritten)");

  virtual
  ~OperationsOsp() = 0;                       // create abstract class

  // PROBLEM BUILDING CALLS (and similar)

public:

  virtual
  void
  uploadShortrunCosts                         // specific costs
  (const CostSet& dutySpecCosts,
   const int      gol);                       // zero means overwrite "shift" term

  virtual
  void
  uploadShortrunCosts                         // specific costs, increment "shift" term
  (const CostSet& shiftCosts);

  // SOLUTION RECOVERY CALLS - general

  virtual
  const double
  downloadSolnVar
  (const int gol) const;

  virtual
  const double
  downloadSolnSlack
  (const int gow) const;

  virtual
  const bool                                  // 'false' means not selected or tripped
  downloadRunStatus
  (const int gol) const;                      // trip global column

}; // class 'OperationsOsp'

// ----------------------------------------------------------
//  CLASS           : QuantityOsp (abstract)
// ----------------------------------------------------------
//  Description  : abstract class covering gateway quantities
//  Role         : parent for concrete "Qan" sub-classes
//  Techniques   : pure virtual destructor
//  Status       : incomplete
// ----------------------------------------------------------

class QuantityOsp :
  public OptimSubProb
{
  // CREATORS

public:

  QuantityOsp
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string&         ospDesc = "(not overwritten)");

  virtual
  ~QuantityOsp() = 0;                         // create abstract class

}; // class 'QuantityOsp'

// ----------------------------------------------------------
//  CLASS           : OfferOsp (abstract)
// ----------------------------------------------------------
//  Description  : abstract class covering gateway offers
//  Role         : parent for concrete "Ofr" sub-classes
//  Techniques   : pure virtual destructor
//  Status       : incomplete
// ----------------------------------------------------------
```

```
class OfferOsp :
  public OptimSubProb
{
  // CREATORS

public:

  OfferOsp
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string&         ospDesc = "(not overwritten)");

  virtual
  ~OfferOsp() = 0;                          // create abstract class

  // MANIPULATORS

public:

  virtual
  boost::tuple
  <int>                                     // sale gol, connects to interface
  uploadTariffSet
  (const shared_ptr<BandedTariffSet> tariffset,
   const double                      capacity) = 0;

}; // class 'OfferOsp'

// ----------------------------------------------------------
//  CLASS           : JunctionOsp (abstract)
// ----------------------------------------------------------
//  Description  : abstract class for supporting demand splitting and joining
//  Role         : parent for concrete "Jnc" sub-classes
//  Techniques   : abstract
//  Status       : complete
//
//  Design notes
//
//      Junctions necessarily lack exergy destruction or cost
//      formation.
//
// ----------------------------------------------------------

class JunctionOsp :
  public OptimSubProb
{
  // CREATORS

public:

  JunctionOsp
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string&         ospDesc = "(not overwritten)");

  virtual
  ~JunctionOsp() = 0;                       // create abstract class

}; // class 'JunctionOsp'

// ----------------------------------------------------------
//  CLASS           : LmpNodeOsp (abstract)
// ----------------------------------------------------------
//  Description  : abstract class for supporting LMP nodes
//  Role         : parent for concrete "Hvn" sub-classes
//  Techniques   : abstract
//  Status       : complete
//
//  Design notes
//
//      LMP nodes necessarily lack exergy destruction or cost
//      formation.
//
// ----------------------------------------------------------

class LmpNodeOsp :
  public OptimSubProb
{
  // CREATORS

public:
```

```
  LmpNodeOsp
  (shared_ptr<svif::SolverIf> solver,
   const xeona::DomainMode    commitmentMode,
   const std::string&         ospDesc = "(not overwritten)");

  virtual
  ~LmpNodeOsp() = 0;                          // create abstract class

}; // class 'LmpNodeOsp'

#endif // _OPTPROB_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : overseer.h
//  file-create-date : Thu 07-Aug-2008 19:53 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : top-level overseer entity (singleton) / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/overseer.h $

//  HEADER GUARD

#ifndef _OVERSEER_H_
#define _OVERSEER_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/costreg.h"     // cost registers

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                               // constructor argument
class DomainController;                     // data member

//  CODE

// ----------------------------------------------------------
//  CLASS           : Overseer
// ----------------------------------------------------------
//  Description  : top level simulation entity
//  Role         : contains 'run' function invoked by free function 'xeona::simulate'
//  Techniques   : unenforced singleton
//  Status       : mostly complete
//
//  Design notes
//
//      Unenforced mandatory singleton
//
//          This is an unenforced singleton, but warnings are
//          issued for multiple instantiations.
//
//          In terms of model requirements, this object is on the
//          mandatory list and must carry the identifier defined
//          by the 'xeona::overseer' string literal in
//          'common.cc' (probably set to "overseer").
//
```

```
//      Ranked originating domain controllers
//
//          First up, domain controllers are also known as
//          "commitment domains" or simply "domains" for short.
//
//          Only the so-called originating domains are specified
//          and then in descending rank order.  The remaining
//          domains are discovered naturally during the domain
//          graph traversal process.
//
//          The domain controllers, in total, form a directed
//          graph.  The various restrictions on this graph are
//          given in my (Robbie Morrison) PhD thesis and are not
//          listed here.
//
// ---------------------------------------------------------

class Overseer :                    // does NOT inherit from class 'TicToc' nor 'Actor'
  public FullEntity,
  public virtual CostRegister,      // provides the set of registers
  public CostRegisterOverseer
{
  // DISABLED

private:

  Overseer();                                       // zero-argument constructor
  Overseer(const Overseer& orig);                   // copy constructor
  Overseer& operator= (const Overseer& orig);       // copy assignment operator

  // CREATORS

public:

  explicit
  Overseer
  (const std::string entityId,
   Record&           record);

  virtual
  ~Overseer();

  // MANIPULATORS

public:

  virtual
  void
  factoryInitialize();                        // called by factory after construction

  // SIMULATION POINT OF ENTRY

public:

  bool
  run
  (const int steps);                          // run with full or restricted horizon

  // INTERNAL DATA

private:

  std::string&                                   d_captrans_algorithm; // selection
  std::string&                                   d_ranked_orig_domains;
  std::vector<shared_ptr<DomainController> >     d_rankedOrigDomains;  // descending rank

  // STATIC DATA

private:

  static unsigned    s_count;                 // instance counter

};

//  ==== XEDOC ==============================================
//
//  entity.overseer
//
//      class                                     > Overseer
//
//        the Overseer entity is REQUIRED and the 'overseer'
```

```
//        identifier is MANDATORY
//
//        the overseer does little more that invoke the various
//        originating domains in nominated order at each new
//        interval
//
//      builtin-remark s                        <
//
//      captrans-algorithm s                    > "simple"
//
//        captrans-algorithm takes 'fixed' | 'simple' | 'hop-relit'
//        but only 'simple' is currently implemented (this call
//        contains experimental macro-controlled hop-relit code)
//
//      ranked-orig-domains L          > "domain-controller-1 domain-controller-2"
//
//        the originating domain controllers must be given in
//        order of DESCENDING priority, any unannounced domains
//        will be discovered naturally during the various
//        traversals -- an originating domain must contain at
//        least one source entity
//
//      total-financial [$] f                   < 0.0
//      total-greenhouse [kg] f                 < 0.0
//      total-nox [kg] f                        < 0.0
//      total-depletion [J] f                   < 0.0
//      total-landuse [m^2] f                   < 0.0
//
//        the cost-type totals cover the entire horizon, with
//        first step truncation given by program.last-run.run-kind
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//      embedded-costs-financial [$] F          < 0.0 ..
//      variable-costs-greenhouse [kg] F        < 0.0 ..
//      fixed-costs-greenhouse [kg] F           < 0.0 ..
//      embedded-costs-greenhouse [kg] F        < 0.0 ..
//      variable-costs-nox [kg] F               < 0.0 ..
//      fixed-costs-nox [kg] F                  < 0.0 ..
//      embedded-costs-nox [kg] F               < 0.0 ..
//      variable-costs-depletion [J] F          < 0.0 ..
//      fixed-costs-depletion [J] F             < 0.0 ..
//      embedded-costs-depletion [J] F          < 0.0 ..
//      variable-costs-landuse [m^2] F          < 0.0 ..
//      fixed-costs-landuse [m^2] F             < 0.0 ..
//      embedded-costs-landuse [m^2] F          < 0.0 ..
//
//  ============================================================

#endif // _OVERSEER_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : propdata.h
//  file-create-date : Thu 05-Feb-2009 11:42 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : mass-based commodity property data / header
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/propdata.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This file declares some commodity properties that should be
//  useful to entity authors.

//  HEADER GUARD

#ifndef _PROPDATA_H_
#define _PROPDATA_H_

//  LOCAL AND SYSTEM INCLUDES

//  no hash-includes required

//  DEFINITIONS: some typical MASS-specific physico-chemical property values

namespace xeona
{
  extern const double massCp_steam;          // constant pressure heat capacity
  extern const double massCp_ice;            // constant pressure heat capacity
  extern const double massCp_liquidWater;    // constant pressure heat capacity

  extern const double massHhv_hardCoal;      // enthalpy of combustion, condensed water
  extern const double massHhv_lignite;       // enthalpy of combustion, condensed water
  extern const double massHhv_natGas;        // enthalpy of combustion, condensed water
  extern const double massHhv_wood;          // enthalpy of combustion, condensed water

  extern const double massCO2_hardCoal;      // carbon-dioxide on oxidation
  extern const double massCO2_lignite;       // carbon-dioxide on oxidation
  extern const double massCO2_natGas;        // carbon-dioxide on oxidation
  extern const double massCO2_wood;          // carbon-dioxide on oxidation

  extern const double massGhg_hardCoal;      // co2-equiv on oxidation, direct
  extern const double massGhg_lignite;       // co2-equiv on oxidation, direct
  extern const double massGhg_natGas;        // co2-equiv on oxidation, direct
  extern const double massGhg_wood;          // co2-equiv on oxidation, direct

  extern const double massGray_hardCoal;     // co2-equiv on oxidation, upstream also
  extern const double massGray_lignite;      // co2-equiv on oxidation, upstream also
  extern const double massGray_natGas;       // co2-equiv on oxidation, upstream also
  extern const double massGray_wood;         // co2-equiv on oxidation, upstream also

} // namespace xeona
```

```
#endif // _PROPDATA_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : registr.h
//  file-create-date : Wed 20-Jun-2007 13:25 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : sub-entity registrations (modify this and 'register.cc') / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/register.h $

//  HEADER GUARD

#ifndef _REGISTR_H_
#define _REGISTR_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"

//  CODE

namespace xeona
{
  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::registerEntyCreators
  // ----------------------------------------------------------
  //  Description  : called by main to register Entity creators
  // ----------------------------------------------------------

  void
  registerEntyCreators();

} // namespace xeona

#endif // _REGISTR_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asset.h
//  file-create-date : Tue 26-Aug-2008 14:15 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : technical asset entity / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas.h $

//  HEADER GUARD

#ifndef _TEAS_H_
#define _TEAS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/tictoc.h"      // inherited interface for entities using common calls
#include "../b/costreg.h"     // cost registers
#include "../b/block.h"       // block entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <vector>             // STL sequence container

//  FORWARD (PARTIAL) DECLARATIONS

class LmpBidSet;                            // member function return
class Record;                               // constructor argument

//  CODE

// ----------------------------------------------------------
//  CLASS           : TechnicalAsset
// ----------------------------------------------------------
//  Description  : technical asset base class
//  Role         : step in the entity inheritance web
//  Techniques   : inheritance
//  Status       : complete
//
//  Design notes
//
//      Cost support
//
//          Cost support comes from whichever cost register is
//          inherited from.
//
//      Ceiling and floor duties
//
//          The data members 'd_ceilingDuty' and 'd_floorDuty'
//          should be reset in concrete technical assets as
//          required.
```

```
//
// -------------------------------------------------------

class TechnicalAsset :
  public Block,
  public TicToc,
  public virtual CostRegister                // provides the set of registers
{
  // DISABLED

private:

  TechnicalAsset();                                          // zero-argument constructor
  TechnicalAsset(const TechnicalAsset& orig);                // copy constructor
  TechnicalAsset& operator= (const TechnicalAsset& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  TechnicalAsset
  (const std::string entityId,
   Record&           record,
   const int         commitmentModeSum);     // passed thru to 'TicToc' constructor

  virtual
  ~TechnicalAsset() = 0;                     // create abstract class

  // ASSET OPERATOR-RELATED ROUTINES

public:

  double                                     // prevailing upper bound on duty
  getCeilingDuty() const;

  double                                     // prevailing lower bound on duty
  getFloorDuty() const;

  void
  setCogenHeatWeight                         // CAUTION: similar class 'AuxHeatLead' calls
  (const double cogenHeatLeadWeight);

  const double
  getCogenHeatWeight() const;                // CAUTION: similar class 'AuxHeatLead' calls

  double                                     // prior duty
  getPriorDuty() const;                      // 'NaN' for step 0

  double                                     // prior size
  getPriorSize() const;                      // 'NaN' for step 0

  virtual
  shared_ptr<BandedTariffSet>                // current tariff set for adaptive behavior
  obtainTariffSet() const;

  // INSTANCE DATA

protected:

  double                      d_ceilingDuty;         // upper capacity in terms of duty
  double                      d_floorDuty;           // lower capacity in terms of duty
  double                      d_cogenHeatLeadWeight; // heat lead weighting [0,1]

};

#endif // _TEAS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas1.h
//  file-create-date : Wed 15-Apr-2009 21:02 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _TEAS1_H_
#define _TEAS1_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/teas.h"        // technical asset entity
#include "../b/costreg.h"     // cost registers

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument
namespace svif { class SolverIf; }            // member function argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

class CmElectricity;

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasHvTransmission
// ----------------------------------------------------------
//  Description  : bi-directional high-voltage transmission
//  Role         : primarily for use in LMP wholesale electricity markets
//  Techniques   : DC power flow, linear discretization for quadratic losses (see OSP)
//  Status       : first-pass complete
//
//  Design notes
//
//      The default flow orientation is specified by the
//      direction from my socket to my cable.
//
//      This asset allows negative flow, unlike most technical
```

```
//      assets.
//
// -------------------------------------------------------

class OpsTransmission_A;                    // necessary for OSP typedef below

class TeasHvTransmission :
  public TechnicalAsset,
  public CostRegisterSRFin,
  public CostRegisterEmbFin
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:

  // CAUTION: note this simple way of swapping OSP
  // implementations, the "_X" postfix needs changing in this one
  // place only -- note also that an implementation change is
  // required to honor the same function names and signatures,
  // but not necessarily the behavior at large

  typedef OpsTransmission_A OpsTransmission; // used for switching implementations

  // DISABLED

private:

  TeasHvTransmission();                                          // zero-argument ctor
  TeasHvTransmission(const TeasHvTransmission& orig);            // copy constructor
  TeasHvTransmission& operator= (const TeasHvTransmission& orig);  // copy assignment oper

  // CREATORS

public:

  explicit
  TeasHvTransmission
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasHvTransmission();

  // CALLS

public:

  virtual void establish() { }              // necessary redefinition

  virtual
  const int                                 // returns zero (no duty coupling)
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }              // necessary redefinition

  // INSTANCE DATA

private:

  // tied quantities

  const double&                         d_injectCapacity;
  const double&                         d_voltage;
  const double&                         d_ohmsPerMetre;
  const double&                         d_length;
  const int&                            d_discretizationSteps;

  const std::string&                    d_gridCommodity;    // 'create' interface calls

  shared_ptr<std::vector<bool> >        d_directions;
```

```
  shared_ptr<std::vector<double> >      d_injections;
  shared_ptr<std::vector<double> >      d_exits;
  shared_ptr<std::vector<double> >      d_relativeDutys;
  shared_ptr<std::vector<double> >      d_relativeLosss;

  shared_ptr<Cable<CmElectricity> >     d_cable;
  shared_ptr<Socket<CmElectricity> >    d_socket;

  // local quantities

  shared_ptr<OpsTransmission>           d_ops;              // specialization required

}; // class 'TeasHvTransmission'

//  ==== XEDOC ================================================
//
//  entity.teas-hv-transmission-0
//
//      class                                 > TeasHvTransmission
//
//        a HV transmission line entity, based on DC power flow
//        and suitable for use under nodal pricing
//
//        my socket label is 'elec-1'
//
//      builtin-remark s                      <
//
//      socket-1 l                            > "node-0.grid-1"
//
//        socket-1 is my node entity
//
//      grid-commodity l                      > "cm-electricity-0"
//
//        grid-commodity defines the underlying commodity
//
//      inject-capacity [W] f                 >  400e+06
//      voltage [V] f                         >  220e+03
//      ohms-per-meter [ohm/m] f              >   30e-06
//      length [m] f                          > 1000e+03
//      discretization-steps [-] i            > 4
//
//        the discretization-steps are the number of steps used
//        to approximate quadratic transmission losses
//
//      directions B                          < 1 ..
//      injections [W] F                      < 0.0 ..
//      exits [W] F                           < 0.0 ..
//      relative-dutys [-] F                  < 0.0 ..
//      relative-losss [-] F                  < 0.0 ..
//
//        directions entry 1 indicates positive flow from my
//        socket to my cable
//
//      nameplate-capacity [W] f              > 400e+06
//      duty-specific-cost-financial [$/J] f  > 100.0
//      size-specific-cost-financial [$/W/s] f  > 200.0
//      standing-cost-financial [$/s] f       > 300.0
//
//      variable-costs-financial [$] F        < 0.0 ..
//      fixed-costs-financial [$] F           < 0.0 ..
//
//      annual-discount-rate-decimal [-] f    > 0.10
//      economic-life [y] i                   > 20
//      capex-initial [$] f                   > 120e+03
//      capex-terminal [$] f                  >  40e+03
//      current-age [y] i                     > 2
//
//        a negative 'capex-terminal' indicates salvage revenue
//        as opposed to decommissioning cost
//
//      embedded-costs-financial [$] F        < 0.0 ..
//
//        the 'embedded-costs-financial' are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//  ============================================================

#endif // _TEAS1_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas02.h
//  file-create-date : Wed 22-Apr-2009 12:40 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 2 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas02.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _TEAS02_H_
#define _TEAS02_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/teas.h"        // technical asset entity
#include "../b/costreg.h"     // cost registers

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                               // constructor argument
namespace svif { class SolverIf; }          // member function argument

template <typename C> class Cable;          // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

class CmElectricity;                        // 'CmElectricity' is from 'Commodity' etc
class CmOxidize;

class CxAmbientAir;

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasOxidToElec
// ----------------------------------------------------------
//  Description  : particularized two-port conversion asset
//  Role         : concrete entity
//  Techniques   : 'OpsFac1Out1'
//  Status       : beta
// ----------------------------------------------------------

class OpsFac1Out1_A;                        // necessary for OSP typedef below

class TeasOxidToElec :
```

```
  public TechnicalAsset,
  public CostRegisterSRFin,
  public CostRegisterEmbFin
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                    // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac1Out1_A OpsFac1Out1;         // used for switching implementations

  // DISABLED

private:

  TeasOxidToElec();                                          // zero-argument constructor
  TeasOxidToElec(const TeasOxidToElec& orig);                // copy constructor
  TeasOxidToElec& operator= (const TeasOxidToElec& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  TeasOxidToElec
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasOxidToElec();

  // CALLS

public:

  virtual
  void
  establish();

  virtual
  const int                                  // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }               // necessary but hollow redefinition

  // INSTANCE DATA

private:

  // tied quantities

  const double&                       d_prodLoBound;
  const double&                       d_prodHiBound;
  const double&                       d_marginalEfficiency;
  const double&                       d_fuelNoload;
  const double&                       d_fuelAncillary;
  const double&                       d_rampRestraintDown;
  const double&                       d_rampRestraintUp;

  shared_ptr<std::vector<double> >    d_productions;
  shared_ptr<std::vector<bool> >      d_shutdownStatuss;

  shared_ptr<Cable<CmOxidize> >       d_inOxid;
  shared_ptr<Socket<CmElectricity> >  d_outElec;

  // local quantities

  shared_ptr<CmOxidize>               d_oxid;
  shared_ptr<OpsFac1Out1>             d_ops;     // specialization required

}; // class 'TeasOxidToElec'
```

```
//  ==== XEDOC ================================================
//
//  entity.teas-oxid-to-elec-0
//
//      class                                   > TeasOxidToElec
//
//        a simple fuel to power (thermal, fuel cell, or
//        otherwise) power plant, with support for shutdown mode
//        operation and ramp rate restrictions -- ancillary
//        electricity demand is not included
//
//        my socket is called elec-1
//
//      builtin-remark s                        <
//
//      socket-oxidize l                        > "teas-supplier-0.oxid-1"
//      cable-oxidize-commodity l               > "cm-oxid-0"
//
//        the socket-oxidize and cable-oxidize-commodity define
//        my supplier and their socket label and our common
//        oxidizable fuel commodity
//
//      socket-electricity-commodity l          > "cm-elec-0"
//
//        the socket-electricity-commodity defines the common
//        electricity commodity
//
//      prod-lo-bound [W] f                     > 60.0
//      prod-hi-bound [W] f                     > 90.0
//      marginal-efficiency [-] f               > 0.5
//      fuel-noload [W] f                       > 10.0
//      fuel-ancillary [W] f                    > 5.0
//
//        the prod-hi-bound and prod-lo-bound set the production
//        capacity and the shutdown mode threshold respectively,
//        the marginal-efficiency and fuel-noload determine the
//        operating curve, and the fuel-ancillary sets the
//        shutdown fuel usage
//
//      ramp-restraint-down [-] f               > 2.0
//      ramp-restraint-up [-] f                 > 2.0
//
//        ramp-restraint-down and ramp-restraint-up restrict
//        output steps relative to nameplate-capacity -- use
//        values of prod-hi-bound/nameplate-capacity or more to
//        prevent binding
//
//      productions [W] F                       < 0.0 ..
//      shutdown-statuss [-] B                  < 0 ..
//
//        productions represent the actual output figures,
//        shutdown-statuss are true (1) if the plant ran
//
//      nameplate-capacity [W] f                > 80.0
//      duty-specific-cost-financial [$/J] f    > 100.0
//      size-specific-cost-financial [$/W/s] f  > 200.0
//      standing-cost-financial [$/s] f         > 300.0
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
//      annual-discount-rate-decimal [-] f      > 0.10
//      economic-life [y] i                     > 25
//      capex-initial [$] f                     > 120e+03
//      capex-terminal [$] f                    > -10e+03
//      current-age [y] i                       > 2
//
//        a negative capex-terminal indicates salvage revenue as
//        opposed to decommissioning cost
//
//      embedded-costs-financial [$] F          < 0.0 ..
//
//        the embedded-costs-financial are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : TeasWindfarm
```

```
// ----------------------------------------------------------
//  Description  : windfarm comprising one or more identical turbines
//  Role         : concrete entity
//  Techniques   : 'OpsFac0Out1' source-style OSP
//  Status       : beta
// ----------------------------------------------------------

class OpsFac0Out1_A;                          // necessary for OSP typedef below

class TeasWindfarm :
  public TechnicalAsset,
  public CostRegisterSRFin,
  public CostRegisterEmbFin
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                     // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac0Out1_A OpsFac0Out1;          // used for switching implementations

  // DISABLED

private:

  TeasWindfarm();                                        // zero-argument constructor
  TeasWindfarm(const TeasWindfarm& orig);                // copy constructor
  TeasWindfarm& operator= (const TeasWindfarm& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  TeasWindfarm
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasWindfarm();

  // CALLS

public:

  virtual
  void
  establish();                                // with low priority development reporting

  virtual
  const int                                   // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual
  void
  conclude();

  // UTILITY FUNCTIONS

protected:

  double                                      // resultant power [W]
  calcTurbinePower                            // single turbine
  (const double windSpeed);                   // known wind speed [m/s]

  // INSTANCE DATA

private:

  // tied quantities
```

```
  assign_ptr<CxAmbientAir>             d_ambientAirContext;

  const int&                           d_count;
  const double&                        d_turbineRating;
  const double&                        d_loCutSpeed;
  const double&                        d_hiCutSpeed;

  shared_ptr<std::vector<double> >     d_potentialProductions;
  shared_ptr<std::vector<double> >     d_actualProductions;
  double&                              d_availability;
  double&                              d_spill;

  shared_ptr<Socket<CmElectricity> >   d_outElec;

  // local quantities

  shared_ptr<OpsFac0Out1>              d_ops;     // specialization required

};

//  ==== XEDOC ================================================
//
//  entity.teas-windfarm-0
//
//      class                                  > TeasWindfarm
//
//        a windfarm comprising one or more identical turbines,
//        which can also spill wind
//
//      builtin-remark s                       <
//
//      count [-] i                            > 5
//      turbine-rating [W] f                   > 1000e+03
//      lo-cut-speed [m/s] f                   > 5.0
//      hi-cut-speed [m/s] f                   > 14.0
//
//        the turbine-rating applies at the cut-out-speed and can
//        be multiplied by count to calculate the windfarm capacity
//
//      socket-electricity-commodity l         > "cm-elec-0"
//
//        socket-electricity-commodity defines the shared
//        electricity commodity
//
//      ambient-air-context l                  > "cx-ambient-air-0"
//
//        the context entity must be a sub-class of CxAmbientAir
//
//      potential-productions [W] F            < 0.0 ..
//      actual-productions [W] F               < 0.0 ..
//      availability [-] f                     < 0.0
//      spill [-] f                            < 0.0
//
//        availability [0,1] is the ratio of potential output to
//        turbine-rating, spill [0,1] is the ratio of discarded
//        to potential output
//
//      nameplate-capacity [W] f               > 5000e+03
//      duty-specific-cost-financial [$/J] f   > 100.0
//      size-specific-cost-financial [$/W/s] f > 200.0
//      standing-cost-financial [$/s] f        > 300.0
//
//      variable-costs-financial [$] F         < 0.0 ..
//      fixed-costs-financial [$] F            < 0.0 ..
//
//      annual-discount-rate-decimal [-] f     > 0.10
//      economic-life [y] i                    > 25
//      capex-initial [$] f                    > +120e+03
//      capex-terminal [$] f                   > -10e+03
//      current-age [y] i                      > 2
//
//        a negative capex-terminal indicates salvage revenues as
//        opposed to decommissioning cost
//
//      embedded-costs-financial [$] F         < 0.0 ..
//
//  ============================================================

#endif // _TEAS02_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas03.h
//  file-create-date : Sat 16-May-2009 12:24 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 3 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas03.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _TEAS03_H_
#define _TEAS03_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/teas.h"        // technical asset entity
#include "../b/costreg.h"     // cost registers

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument
namespace svif { class SolverIf; }            // member function argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

class CmElectricity;                          // 'CmElectricity' is from 'Commodity' etc
class CmOxidize;                              // 'CmOxidize' is from 'Commodity' etc

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasLoadElecTs
// ----------------------------------------------------------
//  Description  : timeseries prescribed electricity load
//  Role         : concrete entity
//  Techniques   : 'OpsFac1Out0' sink-style OSP
//  Status       : first-pass complete
// ----------------------------------------------------------

class OpsFac1Out0_A;                          // necessary for OSP typedef below

class TeasLoadElecTs :
  public TechnicalAsset,
```

```
  public CostRegisterSRFin,
  public CostRegisterEmbFin
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                    // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac1Out0_A OpsFac1Out0;         // used for switching implementations

  // DISABLED

private:

  TeasLoadElecTs();                                          // zero-argument constructor
  TeasLoadElecTs(const TeasLoadElecTs& orig);                // copy constructor
  TeasLoadElecTs& operator= (const TeasLoadElecTs& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  TeasLoadElecTs
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasLoadElecTs();

  // CALLS

public:

  virtual void establish() { }               // necessary redefinition

  virtual
  const int                                  // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }               // necessary redefinition

  // INSTANCE DATA

private:

  // tied quantities

  const shared_ptr<std::vector<double> >    d_loads;
  shared_ptr<Cable<CmElectricity> >         d_inElec;

  // local quantities

  shared_ptr<OpsFac1Out0>                   d_ops;     // specialization required

};

//  ==== XEDOC ================================================
//
//  entity.teas-load-elec-ts-0
//
//      class                                  > TeasLoadElecTs
//
//        prescribed electricity load, lacking both flexibility
//        and context dependency
//
//      builtin-remark s                       <
//
//      socket-electricity 1                   > "teas-supplier-0.elec-1"
//      cable-electricity-commodity 1          > "cm-electricity-0"
//
```

```
//        the above fields define my supplier and their socket
//        label and our common electricity commodity
//
//      loads [W] F                             > 1.0e+06 0.5e+06 ..
//
//        the loads are simply specified
//
//      nameplate-capacity [W] f                > 2.0e+06
//      duty-specific-cost-financial [$/J] f    > 100.0
//      size-specific-cost-financial [$/W/s] f  > 200.0
//      standing-cost-financial [$/s] f         > 300.0
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
//      annual-discount-rate-decimal [-] f      > 0.10
//      economic-life [y] i                     > 25
//      capex-initial [$] f                     > 120e+03
//      capex-terminal [$] f                    > -10e+03
//      current-age [y] i                       > 2
//
//        a negative capex-terminal indicates salvage revenue as
//        opposed to decommissioning cost
//
//      embedded-costs-financial [$] F          < 0.0 ..
//
//        the embedded-costs-financial are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : TeasMineOxid
// ------------------------------------------------------------
//  Description  : oxidizable commodity source
//  Role         : concrete entity
//  Techniques   : 'OpsFac0Out1' source-style OSP
//  Status       : first-pass complete
// ------------------------------------------------------------

class OpsFac0Out1_A;                        // necessary for OSP typedef below

class TeasMineOxid :
  public TechnicalAsset,
  public CostRegisterSRFin
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                   // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac0Out1_A OpsFac0Out1;        // used for switching implementations

  // DISABLED

private:

  TeasMineOxid();                                       // zero-argument constructor
  TeasMineOxid(const TeasMineOxid& orig);               // copy constructor
  TeasMineOxid& operator= (const TeasMineOxid& orig);   // copy assignment operator

  // CREATORS

public:

  explicit
  TeasMineOxid
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasMineOxid();

  // CALLS
```

```
public:

  virtual void establish() { }               // necessary redefinition

  virtual
  const int                                  // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }               // necessary redefinition

  // INSTANCE DATA

private:

  // tied quantities

  const double&                              d_extractLoBound;
  const double&                              d_extractHiBound;

  const shared_ptr<std::vector<double> >     d_extractions;
  shared_ptr<Socket<CmOxidize> >             d_outOxid;

  // local quantities

  shared_ptr<OpsFac0Out1>                    d_ops;     // specialization required

};

//  ==== XEDOC ================================================
//
//  entity.teas-mine-oxid-0
//
//      class                                  > TeasMineOxid
//
//        flexible oxidizable commodity source, currently without
//        an upper bound on capacity
//
//      builtin-remark s                       <
//
//      socket-oxide-commodity l               > "cm-oxidize-0"
//
//        socket-oxide-commodity defines the shared commodity
//
//      extract-lo-bound [J/s] f               > 0.0
//      extract-hi-bound [J/s] f               > 8.0e+03
//
//        extract-hi-bound is required by the simulation whereas
//        nameplate-capacity informs the financial calculations
//
//      extractions [kg] F                     < 0.0 ..
//
//        financial data
//
//      nameplate-capacity [kg/s] f            > 10.0
//      duty-specific-cost-financial [$/kg/s] f  > 100.0
//      size-specific-cost-financial [$/kg/s] f  > 200.0
//      standing-cost-financial [$/s] f        > 300.0
//
//      variable-costs-financial [$] F         < 0.0 ..
//      fixed-costs-financial [$] F            < 0.0 ..
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : TeasMineElec
// ------------------------------------------------------------
//  Description  : electricity commodity source
//  Role         : concrete entity
//  Techniques   : 'OpsFac0Out1' source-style OSP
//  Status       : first-pass complete
// ------------------------------------------------------------

class OpsFac0Out1_A;                         // necessary for OSP typedef below

class TeasMineElec :
  public TechnicalAsset,
```

```
  public CostRegisterSRFin
{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                    // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac0Out1_A OpsFac0Out1;         // used for switching implementations

  // DISABLED

private:

  TeasMineElec();                                      // zero-argument constructor
  TeasMineElec(const TeasMineElec& orig);              // copy constructor
  TeasMineElec& operator= (const TeasMineElec& orig);  // copy assignment operator

  // CREATORS

public:

  explicit
  TeasMineElec
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasMineElec();

  // CALLS

public:

  virtual void establish() { }               // necessary redefinition

  virtual
  const int                                  // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }               // necessary redefinition

  // INSTANCE DATA

private:

  // tied quantities

  const double&                              d_extractLoBound;
  const double&                              d_extractHiBound;

  const shared_ptr<std::vector<double> >     d_extractions;
  shared_ptr<Socket<CmElectricity> >         d_outElec;

  // local quantities

  shared_ptr<OpsFac0Out1>                    d_ops;     // specialization required

};

//  ==== XEDOC ================================================
//
//  entity.teas-mine-elec-0
//
//      class                                     > TeasMineElec
//
//        flexible (and oversimplified) electricity commodity
//        source, currently without an upper bound on capacity --
//        somewhat similar to a battery
//
//      builtin-remark s                          <
//
```

```
//      socket-electricity-commodity l           > "cm-electricity-0"
//
//        socket-electricity-commodity defines the shared
//        commodity
//
//      extract-lo-bound [kJ/s] f                > 0.0
//      extract-hi-bound [kJ/s] f                > 1000.0
//
//        extract-hi-bound is required by the simulation whereas
//        nameplate-capacity informs the financial calculations
//
//      extractions [kJ] F                       < 0.0 ..
//
//        financial data
//
//      nameplate-capacity [kJ/s] f              > 10.0
//      duty-specific-cost-financial [$/kJ/s] f  > 100.0
//      size-specific-cost-financial [$/kJ/s] f  > 200.0
//      standing-cost-financial [$/s] f          > 300.0
//
//      variable-costs-financial [$] F           < 0.0 ..
//      fixed-costs-financial [$] F              < 0.0 ..
//
//  ============================================================

#endif // _TEAS03_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas04.h
//  file-create-date : Thu 08-Oct-2009 12:57 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 4 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas04.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _TEAS04_H_
#define _TEAS04_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/teas.h"        // technical asset entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                               // constructor argument
namespace svif { class SolverIf; }          // member function argument

template <typename C> class Cable;          // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasLoad <>
// ----------------------------------------------------------
//  Description  : operator-coupled load entity
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

template <typename C> class GateCom;        // used by utility functions

class OpsFac1Out0_A;                        // necessary for OSP typedef below

template <typename C>                       // 'C' for commodity
class TeasLoad :
  public TechnicalAsset,
  public CostRegisterSRFin,
```

```
  public CostRegisterEmbFin

{
  // USING DECLARATIONS

protected:

  using Entity::s_logger;                    // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac1Out0_A OpsFac1Out0;         // used for switching implementations

  // DISABLED

private:

  TeasLoad();                                      // zero-argument constructor
  TeasLoad(const TeasLoad& orig);                  // copy constructor
  TeasLoad& operator= (const TeasLoad& orig);      // copy assignment operator

  // CREATORS

public:

  explicit
  TeasLoad
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasLoad();

  // CALLS

public:

  virtual void establish() { }               // necessary redefinition

  virtual
  const int                                  // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }               // necessary redefinition

  // ADAPTIVE SUPPORT

public:

  virtual
  shared_ptr<BandedTariffSet>                // current tariff set for adaptive behavior
  obtainTariffSet() const;

  // UTILITY FUNCTIONS

protected:

  shared_ptr<GateCom<C> >
  obtainGateway() const;

  // INSTANCE DATA

private:

  // tied quantities

  const std::string&                 d_commodity;     // for the 'create' interface calls
  const double&                      d_demandHiBound;

  shared_ptr<std::vector<double> >   d_loads;

  const shared_ptr<Cable<C> >        d_inCommodity;

  // local quantities
```

```
  shared_ptr<OpsFac1Out0>            d_ops;           // specialization required

};

//  ==== XEDOC ========================================================
//
//  entity.teas-load-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//        quantifying extensity * in {J,kg,$}
//
//      class                                  > TeasLoad:Elec
//
//        a load entity which is operator coupled for load
//        definition, for example to AsopInelasticTs
//
//      builtin-remark s                       <
//
//      socket-1 l                             > "teas-supplier-1.elec-1"
//
//        socket-1 is my supplier
//
//      cable-commodity l                      > "cm-electricity-0"
//
//        cable-commodity defines the underlying commodity
//
//      demand-hi-bound [*/s] f                > 5.0e+06
//
//      loads [*/s] F                          < 0.0 ..
//
//        demand-hi-bound is the maximum load, loads are the
//        actual loads
//
//      nameplate-capacity [*/s] f             > 6.0e+06
//      duty-specific-cost-financial [$/*] f     > 100.0
//      size-specific-cost-financial [$/*/s/s] f > 200.0
//      standing-cost-financial [$/s] f          > 300.0
//
//      variable-costs-financial [$] F         < 0.0 ..
//      fixed-costs-financial [$] F            < 0.0 ..
//
//      annual-discount-rate-decimal [-] f       > 0.10
//      economic-life [y] i                      > 25
//      capex-initial [$] f                      > 120e+03
//      capex-terminal [$] f                     > -10e+03
//      current-age [y] i                        > 2
//
//        a negative capex-terminal indicates salvage revenue as
//        opposed to decommissioning cost
//
//      embedded-costs-financial [$] F         < 0.0 ..
//
//        the embedded-costs-financial are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//  ====================================================================

// ------------------------------------------------------------
//  CLASS           : TeasSource <>
// ------------------------------------------------------------
//  Description  : operator-coupled commodity source entity
//  Role         : concrete entity
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

class OpsFac0Out1_A;                        // necessary for OSP typedef below

template <typename C>                       // 'C' for commodity
class TeasSource :
  public TechnicalAsset,
  public CostRegisterSRFin,
  public CostRegisterEmbFin

{
  // USING DECLARATIONS

protected:
```

```
  using Entity::s_logger;                    // place in common scope for this class

  // TYPEDEFS

private:

  typedef OpsFac0Out1_A OpsFac0Out1;         // used for switching implementations

  // DISABLED

private:

  TeasSource();                                          // zero-argument constructor
  TeasSource(const TeasSource& orig);                    // copy constructor
  TeasSource& operator= (const TeasSource& orig);        // copy assignment operator

  // CREATORS

public:

  explicit
  TeasSource
  (const std::string entityId,
   Record&           record);

  virtual
  ~TeasSource();

  // CALLS

public:

  virtual void establish() { }               // necessary redefinition

  virtual
  const int                                  // duty gol
  constrain
  (const xeona::DomainMode capacityMode);

  virtual
  void
  washup();

  virtual void conclude()  { }               // necessary redefinition

  // INSTANCE DATA

private:

  // tied quantities

  const std::string&                 d_commodity;     // for 'create' interface calls
  const double&                      d_extractLoBound;
  const double&                      d_extractHiBound;

  shared_ptr<std::vector<double> >   d_extractions;
  shared_ptr<Socket<C> >             d_outCommodity;

  // local quantities

  shared_ptr<OpsFac0Out1>            d_ops;           // specialization required

}; // class 'TeasSource<C>'

//  ==== XEDOC ==================================================
//
//  entity.teas-source-elec-0
//
//        shortcut documentation
//        base commodity in {Cert,Cseq,Elec,Fund,Heat,Oxid,Thrm,Work}
//        quantifying extensity * in {J,kg,$}
//
//      class                                  > TeasSource:Elec
//
//        an operator-coupled source entity with support for
//        capacity bounds
//
//        my socket is called sock-1
//
//      builtin-remark s                       <
```

```
//
//      socket-commodity l                      > "cm-electricity-0"
//
//        socket-commodity defines the supplied commodity
//
//      extract-lo-bound [*/s] f                > 0.0
//      extract-hi-bound [*/s] f                > 8.0e+06
//
//        extract-hi-bound is used by the simulation whereas
//        nameplate-capacity informs the financial calculations
//
//      extractions [*] F                       < 0.0 ..
//
//        extractions are the actual extractions
//
//      nameplate-capacity [*/s] f              > 6.0e+06
//      duty-specific-cost-financial [$/*] f    > 100.0
//      size-specific-cost-financial [$/*/s/s] f > 200.0
//      standing-cost-financial [$/s] f         > 300.0
//
//      variable-costs-financial [$] F          < 0.0 ..
//      fixed-costs-financial [$] F             < 0.0 ..
//
//      annual-discount-rate-decimal [-] f      > 0.10
//      economic-life [y] i                     > 25
//      capex-initial [$] f                     > 120e+03
//      capex-terminal [$] f                    > -10e+03
//      current-age [y] i                       > 2
//
//        a negative capex-terminal indicates salvage revenue as
//        opposed to decommissioning cost
//
//      embedded-costs-financial [$] F          < 0.0 ..
//
//        the embedded-costs-financial are calculated using the
//        DCF annuity method over the economic life of the entity
//        in question
//
//  ============================================================

#endif // _TEAS04_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teasdev.h
//  file-create-date : Thu 08-Oct-2009 13:00 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets development / header
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teasdev.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  To be used for entity development.

//  HEADER GUARD

#ifndef _TEASDEV_H_
#define _TEASDEV_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/teas.h"        // technical asset entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument
namespace svif { class SolverIf; }            // member function argument

template <typename C> class Cable;            // 'C' is of base class 'Commodity' etc
template <typename C> class Socket;

//  CODE

#endif // _TEASDEV_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : tests.h
//  file-create-date : Thu 24-Jan-2008 15:16 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : test sub-entities for use with --inbuilt and such / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/tests.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Supports the '--inbuilt' command-line option.

//  HEADER GUARD

#ifndef _TESTS_H_
#define _TESTS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../b/entity.h"      // entity base class
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <iostream>           // standard io, used here for overloaded stream inserters
#include <list>               // STL sequence container
#include <string>             // C++ strings
#include <vector>             // STL sequence container

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // see "recset.h"

template <typename C> class Socket;           // see "conex.h"
class CmWork;                                 // see "commods.h"

//  CODE

// ----------------------------------------------------------
//  CLASS           : TestEntity0
// ----------------------------------------------------------
//  Purpose      : an oversimplified first sub-class of FullEntity
//  Status       : working (although difficult to test)
// ----------------------------------------------------------

class TestEntity0 :
  public FullEntity
{
private:

  // DISABLED

  TestEntity0();                                  // prevent zero-argument construction
```

```
  TestEntity0& operator= (const TestEntity0&);    // prevent assignment

public:

  // CREATORS

  TestEntity0
  (const std::string entityId,               // enforced unique identifier
   Record&           record);                // associated record

  virtual                                    // support for polymorphic destruction
  ~TestEntity0();

  void
  factoryInitialize();

}; // class TestEntity0

// ----------------------------------------------------------
//  CLASS           : TestEntity1
// ----------------------------------------------------------
//  Purpose      : a representative first sub-class of FullEntity
//  Status       : working but incomplete
// ----------------------------------------------------------

class TestEntity1 :
  public FullEntity
{
private:

  // DISABLED

  TestEntity1();                                    // prevent zero-argument construction
  TestEntity1& operator= (const TestEntity1&);    // prevent assignment

public:

  // CREATORS

  TestEntity1
  (const std::string entityId,               // enforced unique identifier
   Record&           record);                // associated record

  virtual                                    // support for polymorphic destruction
  ~TestEntity1();

  virtual
  void
  factoryInitialize();

  // STREAM INSERTION SUPPORT

  virtual                                    // CAUTION: needed for polymorphic behavior
  std::ostream&
  streamOut                                  // support for overloaded operator<<
  (std::ostream& os) const;

  // STATIC ACCESSORS

  static
  const std::list<const TestEntity1*>&        // return const reference
  getCensus();

  static
  int
  popnSize();

  static
  std::string
  traverseFullPopulation();

  // UTILITY FUNCTIONS

private:

  // INTERNAL DATA

private:

  // input >
  std::string&                      d_userDescription;
```

```
  double&                           d_x;
  bool&                             d_single;
  shared_ptr<std::vector<int> >     d_timeseries;

  // output <
  shared_ptr<std::vector<int> >     d_electricalOutput;
  double&                           d_runTime;
  shared_ptr<std::vector<bool> >    d_someState;

  // STATIC DATA

private:

  static int                              s_ctorCount;
  static std::list<const TestEntity1*>    s_census;    // list of TestEntity1 instances

}; // class TestEntity1

// ----------------------------------------------------------
//  CLASS           : TestEntity2
// ----------------------------------------------------------
//  Purpose      : a representative first sub-class of FullEntity
//  Status       : working but incomplete
// ----------------------------------------------------------

class TestEntity2 :
  public FullEntity
{
private:

  // DISABLED

  TestEntity2();                                    // prevent zero-argument construction
  TestEntity2& operator= (const TestEntity2&);    // prevent assignment

public:

  // CREATORS

  TestEntity2
  (const std::string entityId,              // enforced unique identifier
   Record&           record);               // associated record

  virtual                                   // support for polymorphic destruction
  ~TestEntity2();

  // MANIPULATORS

  virtual
  void
  factoryInitialize();

  // INTERNAL DATA

private:

  // input >
  // NONE

  // output <
  // NONE

  // other
  shared_ptr<Socket<CmWork> >    d_socketWork;

}; // class TestEntity2

#endif // _TESTS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : tictoc.h
//  file-create-date : Thu 13-Nov-2008 20:26 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : inherited interface for entities using common calls / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/tictoc.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _TICTOC_H_
#define _TICTOC_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../b/optprob.h"     // optimization sub-problem and key sub-classes
#include "../c/costs.h"       // cost sets and support
#include "../c/util2.h"       // free functions which offer general utilities 2

#include <string>             // C++ strings

//  FORWARD (PARTIAL) DECLARATIONS

class CostSet;                                  // "c/costs.h"
namespace svif { class SolverIf; }              // "d/siglp.h"

//  CODE

// ----------------------------------------------------------
//  CLASS           : TicToc
// ----------------------------------------------------------
//  Description  : provide a call interface for common calls
//  Role         : to support a consistent usage across hosts classes
//  Techniques   : multiple inheritance, call to 'xeona::isTwoContained'
//  Status       : incomplete
//
//  CAUTION: see the design notes in the implementation file too.
//
//      This class supplies the protected member functions and data:
//
//          'establish'            beginning of horizon
//          'initialize'           beginning of interval
//          'constrain'            following shortly after 'initialize' call
//          'washup'               end of interval
//          'conclude'             end of horizon
//
```

```
//          'd_step'               current interval count
//          'd_solver'             current solver interface object
//
//      and also supplies and sets the data members:
//
//          'd_commitmentMode'     current commitment mode (aka control strategy)
//          'd_commitmentModeSum'  supported commitment modes
//
//      The clients of this class are as follows:
//
//          currently inherited by:
//
//              'AssetOperator'
//              'Gateway'
//              'TechnicalAsset'
//
//          to be inherited by perhaps:
//
//              'Environment'
//              'Multibus'
//
//          but NOT INHERITED by:
//
//              'DomainController' which implements its own versions
//
//          and NOT REQUIRED by:
//
//              'Overseer' singleton which just contains 'run'
//
// ----------------------------------------------------------

class TicToc
{
  // DISABLED

private:

  TicToc();                                     // zero-argument constructor
  TicToc(const TicToc& orig);                   // copy constructor
  TicToc& operator= (const TicToc& orig);       // copy assignment operator

  // CREATORS

public:

  TicToc
  (const int commstratSum);                     // fixed on construction

  virtual
  ~TicToc();                                    // destructor

  // BEGINNING OF HORIZON CALL - ABSTRACT

  virtual
  void
  establish() = 0;

  // STRUCTURE CHANGE CALL - PRIMARILY DEFINED HERE

  virtual                                       // because 'AssetOperator::restructure' loops
  void
  restructure
  (const xeona::DomainMode commitmentMode);

  // LOOP CALLS - PRIMARILY DEFINED HERE

  virtual                                       // because 'AssetOperator::initialize' loops
  void
  initialize
  (const int                   step,            // CAUTION: the step count is ZERO-based
   shared_ptr<svif::SolverIf> solver);          // solver instance passed thru

  // LOOP CALLS - ABSTRACT

  virtual
  const int                                     // return interpretation is host-dependent
  constrain                                     // set constraints
  (const xeona::DomainMode capacityMode) = 0;

  virtual
  void
```

```
  washup() = 0;

  // END OF HORIZON CALL - ABSTRACT

  virtual
  void
  conclude() = 0;

  // UTILITY FUNCTIONS

protected:

  bool                                      // 'false' is problematic
  checkCommitmentMode                       // non-virtual function
  (const xeona::DomainMode pure,            // value under test
   const xeona::DomainMode sum);            // miscible values aggregate

  // INSTANCE DATA

protected:

  xeona::DomainMode             d_commitmentMode;      // current commitment strategy
  const int                     d_commitmentModeSum;   // supported commitment strategies

  int                           d_step;                // interval count used by clients
  shared_ptr<svif::SolverIf>    d_solver;              // solver object used by clients

  // STATIC DATA

private:

  static logga::spLogger s_logger_tictoc;    // [1] CAUTION

  // [1] alternative logger name: the trailing "tictoc" is to
  // avoid a namespace collision with the logger from the other
  // partner

};

#endif // _TICTOC_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : conex.h
//  file-create-date : Wed 16-Jul-2008 15:22 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : create and connect block interfaces / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/conex.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _CONEX_H_
#define _CONEX_H_

//  AD-HOC NOTES
//
//  See the file 'DOCS/CODE.txt' and, in particular, the section
//  on "Connection terminology" for a review of terms relating to
//  various forms of connections.

//  LOCAL AND SYSTEM INCLUDES

#include "../c/xeona_ptr.h"   // remappable counted pointer which mimics shared_ptr
#include "../b/entity.h"      // entity base class plus lazy linking

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

//  CODE

// ------------------------------------------------------------
//  CLASS           : Interface (abstract base class)
// ------------------------------------------------------------
//  Description  : abstract base class for 'Cable<C>' and 'Socket<C>' class templates
//  Role         : provide common static support for interface instantiations
//  Techniques   : 'weak_ptr', 'shared_ptr' casting, inheritance
//  See also     : 'Cable<C>' and 'Socket<C>'
//  Status       : complete
// ------------------------------------------------------------

// FORWARD (PARTIAL) DECLARATIONS

class Commodity;
class ConnectionOsp;
namespace svif { class SolverIf; }

template <typename C> class Cable;
```

```
template <typename C> class Socket;

// CLASS DECLARATION

class Interface
{
  // DISABLED

private:

  Interface();                                    // zero-argument constructor
  Interface(const Interface& orig);               // copy constructor
  Interface& operator= (const Interface& orig);   // copy assignment operator

  // CREATORS

protected:                                      // access for templated inheritance hierarchy

  Interface
  (const std::string& hostId,                   // host is known at construction time
   const std::string& partnerId,                // empty string for sockets
   const std::string& ifQualifier,              // interface qualifier
   const std::string& commodityId);             // a common commodity is required

protected:

  // PROTECTED VIRTUAL DESTRUCTORS AND SMART POINTERS: first, the
  // use of 'virtual' destructors mean the correct destructors
  // are called.  Second, the use of 'protected' rather than
  // 'public' means that a raw pointer (extracted from a smart
  // pointer, say) cannot use 'delete', but (the original) smart
  // pointer still works fine -- see Karlsson (2006 pp23-24) for
  // an explanation and also Sutter and Alexandrescu (2005 pp90-91).
  //
  // Concrete classes need 'public' destructors and should also
  // be 'virtual' if further derived.

  virtual
  ~Interface() = 0;                             // create abstract class

  // FUNCTIONS FOR OPTIMIZATION SUB-PROBLEMS (OSP)

public:

  bool                                          // 'true' if OSP constraint is completed
  bindOsp                                       // map OSP variables to block interfaces
  (shared_ptr<svif::SolverIf> solver,           // current solver interface instance
   const int                  gol);             // global col from relevant OSP call

  // FUNCTIONS FOR HOST SUPPORT

public:

  shared_ptr<Entity>
  getPartner();                                 // dialog should be via the common commodity

  shared_ptr<Commodity>
  getCm();

//    double&
//    operator()                                // function call operator
//    ()                                        // will needs defining in derived classes
//    { return *d_flow; }

  double&                                       // bind to reference or assign directly
  tieFlow();                                    // CAUTION: overloaded

  void
  tieFlow                                       // CAUTION: overloaded
  (double& flow);                               // set by argument

  void
  sayFlow();                                    // for testing purposes

  void
  sayAll                                        // for testing purposes
  (const std::string& name = "");               // optional name

  // CONNECTIVITY FUNCTIONS -- in order of calling

public:
```

```
  static
  bool
  connectAll();                             // point of entry

private:

  template <typename C>
  static
  shared_ptr<Interface>                     // zero on bad cast, empty on locate fail
  cableCast
  (const shared_ptr<Interface>& interface);

  template <typename C>                     // 'C' is restricted to Commodity sub-classes
  static                                    // NOTE: static
  shared_ptr<Socket<C> >                    // zero on bad cast, empty on locate fail
  locateSocket
  (const std::string& socketKey,            // with/without dot-qualification
   const std::string& cableId);

  template <typename C>                     // remember specialization can be employed
  static
  bool
  makeConnection
  (shared_ptr<Cable<C>  >& cable,
   shared_ptr<Socket<C> >& socket);

  // FUNCTIONS PROVIDED FOR UNIT TESTING

public:

  std::string
  getMyKey();

  bool
  isConnected();

  static
  void
  reset();

  static
  int
  getInterfaceCount();

  static
  int
  getConnectionCount();

  static
  bool
  isComplete();

  // UTILITY FUNCTIONS

private:

  void
  setConnected();

  // INSTANCE DATA

protected:                                  // used by 'Cable<C>' and 'Socket<C>' ctors

  std::string                 d_hostId;         // host block
  std::string                 d_partnerId;      // partner block
  std::string                 d_ifQualifier;    // interface qualifier
  std::string                 d_commodityId;    // associated commodity

  std::string                 d_myKey;          // dot-qualified key for me
  std::string                 d_counterKey;     // dot-qualified key for my counter

private:

  int                         d_gol1;           // duly stored first gol
  bool                        d_connected;      // 'true' if connection made

  shared_ptr<Commodity>       d_commodity;      // linked in commodity as intensities
  shared_ptr<double>          d_flow;           // flow in relevant units as extensity

  assign_ptr<ConnectionOsp>   d_cnn;            // optimization sub-problem
```

```
  // STATIC DATA

private:

  static int                    s_interfaceCount;                // all interfaces
  static int                    s_connectionCount;               // bound interfaces
  static int                    s_connectAllCallCount;

  static std::vector<weak_ptr<Interface> > s_censusCables;       // used for traversals
  static std::vector<weak_ptr<Interface> > s_censusSockets;      // used for traversals

protected:

  static logga::spLogger        s_logger;          // shared_ptr to single logger object

}; // class 'Interface'

// ----------------------------------------------------------
//  CLASS           : Cable <>
// ----------------------------------------------------------
//  Description  : a block-to-block interface is a key 'xeona' abstraction
//  Role         : to provide type-safe connectivity while keeping data overheads low
//  Techniques   : restricted templates, static factory function, no data members
//  Status       : complete
// ----------------------------------------------------------

template <typename C>                       // specializations can also be defined
class Cable :
  public Interface
{
  // DISABLED

private:

  Cable();                                  // zero-argument constructor
  Cable(const Cable& orig);                 // copy constructor
  Cable& operator= (const Cable& orig);     // copy assignment operator

  // CREATORS - with private constructor accessed via public static factory function

private:

  Cable
  (const std::string& hostId,               // described for factory function (below)
   const std::string& partnerId,
   const std::string& ifQualifier,
   const std::string& commodityId);

public:

  static                                    // NOTE: static factory function
  shared_ptr<Cable<C> >
  create                                    // sub-id form (added later)
  (const std::string& hostId,               // me
   const std::string& targetId_ifQualifier, // form with dot-separated socket qualifer
   const std::string& commodityId);         // associated commodity

}; // class 'Cable<>'

// ----------------------------------------------------------
//  CLASS           : Socket <>
// ----------------------------------------------------------

template <typename C>                       // specializations can also be defined
class Socket :
  public Interface
{
  // DISABLED

private:

  Socket();                                 // zero-argument constructor
  Socket(const Socket& orig);               // copy constructor
  Socket& operator= (const Socket& orig);   // copy assignment operator

  // CREATORS - with private constructor accessed via public static factory function

private:

  Socket
```

```
   (const std::string& hostId,                // described for factory function (below)
    const std::string& partnerId,
    const std::string& ifQualifier,
    const std::string& commodityId);

public:

   static                                     // NOTE: static factory function
   shared_ptr<Socket<C> >
   create
   (const std::string& hostId,                // me
    const std::string& ifQualifier,           // my socket qualifier
    const std::string& commodityId);          // associated commodity

}; // class 'Socket<>'

#endif // _CONEX_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : costs.h
//  file-create-date : Fri 17-Oct-2008 12:00 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : cost sets and support / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/costs.h $
//
//  GENERAL NOTES FOR THIS FILE

//  AD-HOC NOTES
//
//  Code and unit tests for class 'CostSetR' can be found in
//  r2160.  This material has subsequently been deleted.

//  HEADER GUARD

#ifndef _COSTS_H_
#define _COSTS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/tuple/tuple.hpp>        // n-tuples, ref(), cref()

//  CODE

// ------------------------------------------------------------
//  TYPEDEF         : tupleD5
// ------------------------------------------------------------

typedef boost::tuple<double, double, double, double, double> tupleD5;

// ------------------------------------------------------------
//  CLASS           : CostSet
// ------------------------------------------------------------
//  Description  : container for multiple cost types, be they intensive or extensive
//  Role         : to support 'OptimSubProb' instances and for cost aggregation
//  Techniques   : struct, overloaded operators, tuples
//  Status       : complete
//
//  Design notes
//
//      Cost sets support the (currently five) literal cost types
//      as public data members.
//
//      Cost sets can be used in several contexts to hold:
```

```
//
//         - cost formation parameters - namely specific costs
//
//         - accrued costs - or their various contributions such as
//             - variable
//             - fixed
//             - embedded
//
//        - unit costs - the cost of unit output
//             - average
//             - marginal
//
//      Cost sets do not apply to prices, which are limited to
//      the financial cost type and often held as 'double's.
//      That said, there is no reason preventing incomplete cost
//      sets being used to hold prices.
//
//  Further information
//
//      Please refer to my (Robbie Morrison) PhD thesis for more
//      information.
//
// ---------------------------------------------------------

class CostSet
{
  // CAUTION: the various overloaded free function operators do
  // not need to be granted friendship as the data members are
  // public

  // DISABLED

  // none

  // CREATORS

public:

  CostSet                                   // also acts as a zero argument constructor
  (const double init = 0.0);

  CostSet                                   // up-front constructor
  (const double a_fin,
   const double a_ghg,
   const double a_nox,
   const double a_dep,
   const double a_luc);

  CostSet                                   // tuple constructor
  (const tupleD5 tup);

  ~CostSet();                               // destructor

  // UNARY OPERATORS

  CostSet  operator- () const;                    // unary minus (not subtraction)
  CostSet& operator+= (const CostSet& other);
  CostSet& operator-= (const CostSet& other);
  CostSet& operator*= (const double&  other);     // common multiplier
  CostSet& operator*= (const tupleD5& other);     // supports distinct multipliers

  // ACCESSORS

  // NOTE: the data members are also public and directly accessible

  tupleD5 tuple()  const;                   // export 'CostSet' to five tuple
  bool    isZero() const;                   // 'true' if all entries are zero

  // MANIPULATORS

  void reset(const double value = 0.0);     // reset all values, note the default of zero

  void setFin(const double setFin);         // equivalent to: cs.fin = setFin
  void setGhg(const double setGhg);
  void setNox(const double setNox);
  void setDep(const double setDep);
  void setLuc(const double setLuc);

  void plusFin(const double plusFin);       // equivalent to: cs.fin += plusFin
  void plusGhg(const double plusGhg);
  void plusNox(const double plusNox);
```

```
  void plusDep(const double plusDep);
  void plusLuc(const double plusLuc);

  void multFin(const double multFin);        // equivalent to: cs.fin *= multFin
  void multGhg(const double multGhg);
  void multNox(const double multNox);
  void multDep(const double multDep);
  void multLuc(const double multLuc);

  // DISPLAY CALLS

  std::string
  summarizeMeG                               // general float format using %10.3g
  (std::string msg = "") const;

  std::string
  summarizeMeF                               // fixed float format using %10.2f
  (std::string msg = "") const;

  // STREAM INSERTION SUPPORT

  std::ostream&
  streamOut                                  // support for overloaded operator<<
  (std::ostream& os) const;

  // INSTANCE DATA

public:                                      // CAUTION: public access is correct

  double    fin;                             // financial cost type
  double    ghg;                             // greenhouse gas cost type
  double    nox;                             // NOx cost type
  double    dep;                             // depletable resource use cost type
  double    luc;                             // land use cost type

  // STATIC DATA

private:

  static logga::spLogger    s_logger;        // shared_ptr to single logger object

};

// ------------------------------------------------------------
//  FREE FUNCTION   : operator<< (std::ostream&, CostSet&)
// ------------------------------------------------------------
//
//  CAUTION: place after client class declaration
//
// ------------------------------------------------------------

inline                                       // CAUTION: inline essential
std::ostream&
operator<<                                   // overloaded operator
(std::ostream& os,
const CostSet& cs)
{
  return cs.streamOut(os);                   // wrapper to 'streamOut' member function
}

// FREE FUNCTION BINARY OPERATORS - arithmetic, comparison

// ------------------------------------------------------------
//  FREE FUNCTION   : operator+ (CostSet&, CostSet&)
//  FREE FUNCTION   : operator- (CostSet&, CostSet&)
//  FREE FUNCTION   : operator* (double&,  CostSet&)
//  FREE FUNCTION   : operator* (CostSet&, double&)
//  FREE FUNCTION   : operator* (CostSet&, tupleD5&)
//  FREE FUNCTION   : operator* (tupleD5&, CostSet&)
// ------------------------------------------------------------

const CostSet operator+ (const CostSet& lhs, const CostSet& rhs);
const CostSet operator- (const CostSet& lhs, const CostSet& rhs);
const CostSet operator* (const double&  lhs, const CostSet& rhs);
const CostSet operator* (const CostSet& lhs, const double&  rhs);
const CostSet operator* (const CostSet& lhs, const tupleD5& rhs);
const CostSet operator* (const tupleD5& lhs, const CostSet& rhs);

// ------------------------------------------------------------
//  FREE FUNCTION   : operator== (CostSet&, CostSet&)
//  FREE FUNCTION   : operator!= (CostSet&, CostSet&)
```

```
// -----------------------------------------------------------

bool operator== (const CostSet& lhs, const CostSet& rhs);
bool operator!= (const CostSet& lhs, const CostSet& rhs);

// -----------------------------------------------------------
//  FREE FUNCTION   : operator- (boost::tuple<CostSet, CostSet>)
// -----------------------------------------------------------
//  Description  : unary minus (not subtraction)
//  Role         : intended for turning costs into revenues thru sign-change
//  Techniques   : (nothing special)
//  Status       : incomplete
// -----------------------------------------------------------

boost::tuple <CostSet, CostSet>              // normally "var" and "fix" costs
operator-                                    // unary minus (not subtraction)
(const boost::tuple <CostSet, CostSet>& other);

#endif // _COSTS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : datio.h
//  file-create-date : Mon 01-Oct-2007 12:24 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : model file read, process, and write functionality / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/datio.h $

//  HEADER GUARD

#ifndef _DATIO_H_
#define _DATIO_H_

//  LOCAL AND SYSTEM INCLUDES

#include "recset.h"           // record set support
#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <vector>             // STL sequence container

#include <boost/any.hpp>                // safe type heterogeneous storage
#include <boost/filesystem.hpp>         // path objects, iterators, useful operations
#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  CODE

// ----------------------------------------------------------
//  CLASS           : DataIo
// ----------------------------------------------------------
//  Description  : management of model data input and output
//  Role         : initially read and later overwrite the XEM model file
// ----------------------------------------------------------

class DataIo
{
  // LOCAL ENUMERATIONS

private:

  enum FieldWriteDepth                      // controls field writing
    {
      e_notSet        = 0,                  // set on construction
      e_rawString     = 1,                  // write simply echos back the raw string
      e_boundValue    = 2                   // write unlex the bound values
    };
```

```
  // DISABLED

private:

  DataIo();                                  // zero-argument constructor
  DataIo(const DataIo& orig);                // copy constructor
  DataIo& operator= (const DataIo& orig);    // copy assignment operator

  // CREATORS

public:

  DataIo
  (RecordSet& recset);                       // need to explicitly supply a record-set

  ~DataIo();                                 // destructor

  // ACCESSORS

  boost::filesystem::path                    // complete (absolute) path
  getModelFilePath();

  unsigned
  getRecordCount();

  // STATIC ACCESSORS

  static
  unsigned
  getTabset();

  // MANIPULATORS (identifiers contain the phrase "model")

  // normally called in the order given

  void
  readModel
  (boost::filesystem::path modelFile);       // model file path

  bool                                       // 'true' is model format and svn rev align
  confirmModelSvnAlign();

  unsigned                                   // returns horizon -- or zero on failure
  locateModelHorizon();                      // looks for 'steps' in 'time-horizon'

  void
  processModel();                            // turn raw strings into split strings

  void
  noteModelIsBound();

  void
  updateModelRunTime
  (const std::string& currentSimRet,
   const std::string& currentSimKind);

  std::vector<shared_ptr<Record> >           // can be an empty vector
  getSubset                                  // subset based on record kind
  (const xeona::RecordKind recKind);         // must come after 'processModel'

  void
  writeModel                                 // defaults to empty path for 'stdout'
  (boost::filesystem::path modelFile = boost::filesystem::path());

  void
  setFieldWriteDepth
  (FieldWriteDepth fwd);                     // normally reset to 'DataIo::e_rawString'

  // UTILITY FUNCTIONS (identifiers often contain the phrase "data")

private:

  bool                                       // returns 'true' on success
  loadDataLine
  (const char* cline);                       // C-string input line (from getline)

  bool
  updateProgramOutput
  (const std::string& recordName,            // sought record name
   const std::string& fieldName,             // sought field name
```

```
   const std::string& overwriteStr);         // overwrite if field exists and is output

  void
  writeData                                  // used by 'writeModel'
  (std::ostream& os);

  std::string
  recordKindToLead
  (xeona::RecordKind kind) const;

  std::string
  getFieldValue
  (const shared_ptr<Field> f,
   FieldWriteDepth         fwd = e_notSet);  // default given

  void
  modifyEmacsTabstops
  (std::string& commentLine) const;

  // INTERNAL DATA

private:

  RecordSet&                 d_records;
  boost::filesystem::path    d_model;             // model file complete path

  shared_ptr<Record>         d_curRecord;         // current record
  shared_ptr<Field>          d_curField;          // current field

  FieldWriteDepth            d_fieldWriteDepth;   // controls how field values are derived
  unsigned                   d_lineCount;         // model file line count
  unsigned                   d_recCount;          // pushed-back record count
  bool                       d_flag_initialRecord;
  bool                       d_flag_endMarker;

  // STATIC DATA

private:

  static unsigned            s_tabset;            // tab setting for aligning "<>" output
  static logga::spLogger     s_logger;            // shared_ptr to single logger object

};

#endif // _DATIO_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : extunits.h
//  file-create-date : Wed 02-Dec-2009 11:24 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : quantifying extensity enums and interpretation / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/extunits.h $

//  HEADER GUARD

#ifndef _EXTUNITS_H_
#define _EXTUNITS_H_

//  LOCAL AND SYSTEM INCLUDES

#include <string>             // C++ strings

//  CODE

namespace xeona
{

  // ------------------------------------------------------------
  //  ENUM            : xeona::ExtensityUnit
  // ------------------------------------------------------------

  // CAUTION: these enums are not proceeded by "e_" in the
  // interests of brevity

  enum ExtensityUnit
    {
      notSpecified   = 0,
      joule          = 1,
      kilogram       = 2,
      uoa            = 3,
      metreSq        = 4
    };

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::interpretExtensity
  // ------------------------------------------------------------

  std::string
  interpretExtensity
  (const xeona::ExtensityUnit extensity);

} // namespace 'xeona'

#endif // _EXTUNITS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : factory.h
//  file-create-date : Tue 22-May-2007 12:58 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : sub-entity factory / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/factory.h $

//  HEADER GUARD

#ifndef _FACTORY_H_
#define _FACTORY_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../b/register.h"    // sub-entity registrations
#include "../a/exapp.h"       // application exception classes
#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // switch easily between TR1 and Boost smart pointers

#include <string>             // C++ strings
#include <map>                // STL associative container

//  FORWARD (PARTIAL) DECLARATIONS

class Entity;                                 // from "entity.h"
class Record;                                 // from "recset.h"

//  CODE

// ----------------------------------------------------------
//  CLASS           : EntityFactory
// ----------------------------------------------------------
//  Description  : an enum-keyed entity object factory
//  Patterns     : concrete object factory, singleton
//  Status       : complete
//
//  Design notes
//
//      Note the earlier typedef: "shared_ptr<Entity> entity_type"
//
// ----------------------------------------------------------

class EntityFactory
{
  // TYPEDEFS

public:

  typedef shared_ptr<Entity> (*createEntityCallback)(const std::string&, Record&);

private:
```

```
  typedef shared_ptr<EntityFactory> entityFactory_type;
  typedef std::map<std::string, createEntityCallback> callback_map;

  // DISABLED

private:

  EntityFactory();                                          // zero-argument constructor
  EntityFactory(const EntityFactory& other);                // copy constructor
  EntityFactory& operator= (const EntityFactory& other);    // assignment operator

  // CREATORS

public:

  ~EntityFactory();      // CAUTION: shared_ptrs normally require a public destructor

  // SINGLE POINT OF ACCESS

public:

  static
  entityFactory_type
  iface();

  // MANIPULATORS

public:

  // NOTE: the term 'entity' in function names implies sub-classes too

  bool                                      // true if successful
  registerEntity                            // used in 'register.cc'
  (const std::string    entityRegn,         // string defined in "register.h"
   createEntityCallback createfn);

  bool                                      // true if previously registered
  unregisterEntity
  (const std::string entityRegn);

  shared_ptr<Entity>
  createEntityBind                          // note 'initialize()' member function
  (const std::string  entityRegn,
   const std::string& entityId,             // enforced unique identifier
   Record&            r)                    // associated record to bind
    throw(std::exception,                   // exception specification
          xeona::empty_wrap,
          xeona::non_registration);

  // INTERNAL DATA

private:

  callback_map    d_callbacks;     // std::map<std::string, createEntityCallback>

  // STATIC DATA

private:

  static entityFactory_type    s_instance;   // pointer to this singleton
  static logga::spLogger       s_logger;     // shared_ptr to single logger object

};

#endif // _FACTORY_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : files.h
//  file-create-date : Fri 09-Nov-2007 13:38 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions for regular files / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/files.h $

//  GENERAL NOTES FOR THIS FILE
//
//  The functions here are (inadvertent mistakes aside)
//  POSIX-compliant.  They should therefore be acceptable on
//  Windows (Win32 API and better) as well as on UNIX and Linux.

//  HEADER GUARD

#ifndef _FILES_H_
#define _FILES_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/filesystem.hpp>         // path objects, iterators, useful operations

#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  CODE

namespace xeona
{
  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::testWritable
  // ----------------------------------------------------------
  //  Description  : check if a given Boost.Filesystem path is writable
  // ----------------------------------------------------------

  tribool                                 // indeterminate means status not resolved
  testWritable
  (boost::filesystem::path regularFile);     // this function assumes the file is regular

  // ----------------------------------------------------------
```

```
  //  FREE FUNCTION   : xeona::establishModelFile
  // ------------------------------------------------------
  //  Description  : build path and confirm model file or show failure
  // ------------------------------------------------------

  boost::filesystem::path                   // empty path indicates failure
  establishModelFile                        // assemble path and confirm model file
  (const std::string& modelStub);           // from command-line or "" to use default

  // ------------------------------------------------------
  //  FREE FUNCTION   : xeona::readonly
  // ------------------------------------------------------
  //  Description  : chmod given 'filename' to 'newMode' (0440 set in implementation)
  //  Role         : general use (but often after various GLPK file creation calls)
  //  Techniques   : POSIX 'chmod'
  //  Status       : complete
  // ------------------------------------------------------

  bool                                      // 'false' if unsuccessful
  readonly                                  // wrapper function
  (const char* filename);

  bool                                      // 'false' if unsuccessful
  readonly                                  // principal call
  (const std::string& filename);

} // namespace xeona

#endif // _FILES_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : fincalc.h
//  file-create-date : Fri 17-Oct-2008 14:58 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : support for discounted cash flow analysis / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/fincalc.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  The implementation file contains a full description of the
//  formulas used and similar matters.

//  HEADER GUARD

#ifndef _FINCALC_H_
#define _FINCALC_H_

//  LOCAL AND SYSTEM INCLUDES

// none required

//  CODE

namespace xeona
{

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::presentValue
  // ----------------------------------------------------------

  double
  presentValue                // PV, year-zero present value
  (double futureValue,        // FV, projected future value
   int    timePeriod,         // t, annual with zero-based count
   double interestRate);      // i, interest rate per annum and suitably risk-adjusted

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::annuityValue
  // ----------------------------------------------------------

  double
  annuityValue                // A, annuity value
  (double   presentValue,     // PV, year-zero present value
   unsigned paymentsCount,    // n, number of annual payments
   double   interestRate);    // i, interest rate per annum and suitably risk-adjusted

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::capitalRecovery
  // ----------------------------------------------------------
  //  Description  : calculates the capital recovery
  //  Role         : used for accrued costs reporting
```

```
  //  Techniques   : uses 'presentValue' and 'annuityValue' functions
  //  Status       : complete
  // ------------------------------------------------------------

  double
  capitalRecovery                 // relative to the length of the time horizon interval
  (double   discountRate,         // risk-adjusted interest per annum (decimal not %age)
   unsigned economicLife,         // economic life in years (not seconds)
   double   capitalInputInitial,  // capital input at beginning of economic life
   double   capitalInputTerminal, // capital input at end of economic life (positive if a
   unsigned currentAge = 0);      //                                          liability)

} // namespace xeona

#endif // _FINCALC_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : greenhouse.h
//  file-create-date : Mon 12-Oct-2009 09:10 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : global warming potential support / heater
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/ghouse.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _GHOUSE_H_
#define _GHOUSE_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/tuple/tuple.hpp>        // n-tuples, ref(), cref()

//  CODE

// ------------------------------------------------------------
//  TYPEDEF         : tupleD3
// ------------------------------------------------------------

typedef boost::tuple<double, double, double> tupleD3;

// ------------------------------------------------------------
//  CLASS           : Gwp100Bundle
// ------------------------------------------------------------
//  Description  : container for a global warming potential (GWP) gas bundle
//  Role         : to support 'ghg' cost establishment
//  Techniques   : struct, tuples
//  Status       : complete
//
//  Modeling issues
//
//      Carbon dioxide equivalent (CO2-e)
//
//          CO2-e here represents just three common greenhouse
//          gases weighted according to their 100-year global
//          warming potential (GWP):
//
//              gas               100-year GWP
//              ---------------------------------
//              CO2  carbon dioxide       unity
```

```
//              CH4  methane                25
//              N2O  nitrous oxide         298
//              -------------------------------
//              source: IPCC AR4 WG1 (Forster etal 2007)
//
//      Biofuels
//
//          In the case of biofuels, the sequestered CO2-e
//          inventory as a result of cropping can be subtracted,
//          given that the cropping regime did not necessitate
//          natural ecosystem conversion. However, any remote
//          joint-implementation greenhouse gas mitigation
//          exercise, including carbon sinking, associated with
//          particular assets should not be accounted within a
//          given domain, unless irrevocably linked. Ultimately,
//          decisions relating to CO2-e inclusion or exclusion
//          reside with the modeler.
//
//  Design notes
//
//      Class 'Gwp100Bundle' can be be used in two contexts (in a
//      manner similar to the usage of class 'CostSet'):
//
//          - the mass-specific GWP of a given gas bundle
//          - the carbon dioxide equivalent a given release
//
//  References
//
//      Forster, P, V Ramaswamy, etal.  2007.  Changes in
//        atmospheric constituents and in radiative forcing.
//        In: Climate change 2007 -- the physical science basis :
//        contribution of Working Group I to the Fourth
//        Assessment Report of the Intergovernmental Panel on
//        Climate Change.  Cambridge University Press, Cambridge,
//        United Kingdom and New York, NY, USA.
//
// -----------------------------------------------------------

class Gwp100Bundle
{
  // DISABLED

  // none

  // CREATORS

public:

  Gwp100Bundle();                           // also acts as a zero argument constructor

  Gwp100Bundle                              // up-front constructor
  (const double a_co2,                      // carbon dioxide
   const double a_ch4,                      // methane
   const double a_n2o);                     // nitrous oxide

  Gwp100Bundle                              // tuple constructor
  (const tupleD3 tup);

  ~Gwp100Bundle();                          // destructor

  // UNARY OPERATORS

  Gwp100Bundle& operator*= (const double&  other);     // common multiplier

  // ACCESSORS

  // NOTE: the data members are also public and directly accessible

  tupleD3 tuple() const;                    // export 'Gwp100Bundle' to six tuple

  double totalMass() const;
  double co2Equivalent() const;
  double gwpEffective() const;

  // STATIC ACCESSORS

  double static getGwpCo2();
  double static getGwpCh4();
  double static getGwpN2o();

  // MANIPULATORS
```

```
  void reset(const double value = 0.0);      // reset all values, note the default of zero

  void setCo2  (const double setCo2);        // equivalent to: gwp.co2 = setCo2
  void setCh4  (const double setCh4);
  void setN2o  (const double setN2o);

  // DISPLAY CALLS

  std::string
  summarizeMe
  (std::string msg = "") const;

  static
  std::string
  displayGwps();

  // INSTANCE DATA

public:                                      // CAUTION: public access is correct

  double    co2;                             // carbon dioxide
  double    ch4;                             // methane
  double    n2o;                             // nitrous oxide

  // STATIC DATA

private:

  static const double       s_gwpCo2;        // CO2 global warming potential (unity)
  static const double       s_gwpCh4;        // CH4 global warming potential
  static const double       s_gwpN2o;        // N2O global warming potential

  static logga::spLogger    s_logger;        // shared_ptr to single logger object

};

// ------------------------------------------------------------
//  FREE FUNCTION   : operator* (const double&,       const Gwp100Bundle&);
//  FREE FUNCTION   : operator* (const Gwp100Bundle&, const double&);
// ------------------------------------------------------------

const Gwp100Bundle operator* (const double&       lhs, const Gwp100Bundle& rhs);
const Gwp100Bundle operator* (const Gwp100Bundle& lhs, const double&       rhs);

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::totalMass
//  FREE FUNCTION   : xeona::co2Equivalent
//  FREE FUNCTION   : xeona::gwpEffective
// ------------------------------------------------------------

namespace xeona
{
  double
  totalMass(const double co2,
            const double ch4   = 0.0,
            const double n2o   = 0.0);

  double
  co2Equivalent(const double co2,
                const double ch4   = 0.0,
                const double n2o   = 0.0);

  double
  gwpEffective(const double co2,
               const double ch4   = 0.0,
               const double n2o   = 0.0);
} // namespace 'xeona'

#endif // _GHOUSE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : inbuilt.h
//  file-create-date : Mon 14-Jan-2008 15:28 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : compiled-in test file generation / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/inbuilt.h $

//  HEADER GUARD

#ifndef _INBUILT_H_
#define _INBUILT_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <fstream>            // file-based io
#include <string>             // C++ strings

//  CODE

namespace xeona
{
  std::string                                 // often leaf but directory part will remain
  createModelName
  (const std::string& stub);                  // empty string uses inbuilt default

  bool                                        // 'false' means file open failed
  dumpToFile                                  // calls 'loadOstream' internally
  (const std::string& filename,               // will overwrite file, maybe with zero bytes
   const unsigned     steps);                 // horizon steps

  void
  loadOstream                                 // helper func, contains model in text form
  (std::ostream&  os,
   const unsigned steps);                     // horizon steps

} // namespace xeona

#endif // _INBUILT_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : label.h
//  file-create-date : Fri 24-Oct-2008 12:18 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : helper class for solver labels  / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/label.h $

//  HEADER GUARD

#ifndef _LABEL_H_
#define _LABEL_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

//  CODE

// ----------------------------------------------------------
//  CLASS           : Label
// ----------------------------------------------------------
//  Description  : helper class to collect and format OSP and solver labels
//  Role         : simplify label processing in OSP upload calls
//  Techniques   : overloaded function call operator, output streaming
//  Status       : complete
//
//  Example
//
//      Label lab1;                         // no data
//
//      int a = 4;
//      Label lab2("yes");                  // data added
//      lab2 << "hello";
//      lab2 << "world";
//      lab2 << a;                          // earlier set to 4
//      std::cout << "lab2 (as string) : " << lab2.str()        << std::endl;
//      std::cout << "lab2 (streaming) : " << lab2              << std::endl;
//      std::cout << "lab2 (extra)     : " << lab2.str("extra") << std::endl;
//
//      lab2 (as string) : yes-hello-world-4
//      lab2 (streaming) : yes-hello-world-4
//      lab2 (extra)     : yes-hello-world-4-extra
//
//  CAUTION: stream chaining
//
//      Chaining is not (currently) supported, for instance the
```

```
//      following will not compile:
//
//          lab1 << "first" << "second";
//
//  See also
//
//      See the unit test file for more examples.
//
// ----------------------------------------------------------

class Label
{
  // FRIENDS

  friend std::ostream& operator<< (std::ostream& os, const Label& label);

  // CREATORS

public:

  Label
  (const std::string& str = "");

  ~Label();

  // PUBLIC CALLS

public:

  void
  trim                                      // multiple pop without returning elements
  (int num = 1);                            // defaults to one pop

  void
  resetToCtor();                            // reset to construction-time state

  // CAUTION: the 'add' function template is defined in this
  // header to facilitate implicit template instantiation

  template <typename T>
  bool                                      // 'true' if added
  add
  (const T& in)
  {
    std::ostringstream oss;
    oss << std::boolalpha;                  // modify the default ios flags
    oss << in;                              // stream the input
    const std::string sin = oss.str();
    if ( sin.empty() ) return false;
    d_labelettes.push_back(sin);
    return true;
  }

  bool                                      // 'true' if added
  padd
  (const unsigned in,
   const unsigned pad = 2);                 // pad level, note the general default

  // CAUTION: the 'operator<<' function template is defined in
  // this header to facilitate implicit template instantiation
  //
  // CAUTION: regarding 'operator<<', see the implementation file
  // for explicit specializations covering 'int' and 'unsigned'
  // -- these call 'padd(in)' instead

  template <typename T>
  void
  operator<<                                // streams as per any default ostream
  (const T& in)
  {
    add(in);
  }

  // CAUTION: the 'str(in)' function template is defined in this
  // header to facilitate implicit template instantiation

  template <typename T>
  std::string
  str                                       // stringify with temporary 'in' added
  (const T& in)                             // CAUTION: 'in' is not stored
  {
```

```
    const bool added   = add(in);            // element inserted
    std::string buffer = str();              // overloaded member function 'str()'
    if ( added ) trim(1);                    // element removed
    return buffer;
  }

  std::string
  padstr                                     // zero-pad the integer
  (const unsigned in,
   const unsigned pad = 2);                  // pad level, note default

  std::string
  str() const;                               // principal stringify function

  bool                                       // 'true' if empty
  empty() const;

  // INSTANCE DATA

private:

  const std::string           d_separator;   // separator string (a dot or underscore?)
  const std::string           d_ctorArg;     // argument supplied at construction-time
  std::vector<std::string>    d_labelettes;  // stack of substrings

};

// STREAMING SUPPORT

// ------------------------------------------------------------
//  FREE FUNCTION   : operator<< (std::ostream&, Label&)
// ------------------------------------------------------------

std::ostream&
operator<<
(std::ostream& os,
 const Label&  label);

#endif // _LABEL_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : linklog.h
//  file-create-date : Thu 30-Jul-2009 21:25 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : utility class to record entity linking results / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/linklog.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _LINKLOG_H_
#define _LINKLOG_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/xeona_ptr.h"   // remappable counted pointer which mimics shared_ptr
#include "../c/util1.h"       // free functions which offer general utilities 1

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <typeinfo>           // run-time type information (RTTI)
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>        // n-tuples, ref(), cref()

//  FORWARD (PARTIAL) DECLARATIONS

class Entity;

//  CODE

// ----------------------------------------------------------
//  CLASS           : LinkLogger
// ----------------------------------------------------------
//  Description  : small utility class to recover and hold entity linking details
//  Role         : used by static function 'Entity::linkAll' to create record
//  Techniques   : Boost.Tuple library, Boost.Format library, 'typeid'
//  Status       : complete
//
//  Design notes
//
//      This class represents a central place for recording
//      lazy-to-full link call details for later display.
//
//      The object is held as:
//
```

```
//          class Entity
//          private: static LinkLogger s_linkLog
//
//      This class does not report to the console.
//
//  CAUTION: tuple member function 'get<0>' not g++ 4.1.2
//
//      See note in code regarding this call.
//
// ----------------------------------------------------------

class LinkLogger
{
  // TYPEDEFS

private:

  typedef boost::tuple<std::string,               // entity identifier
                       std::string,               // entity type on construction
                       std::string,               // requested type cast on linking
                       std::string,               // raw address (zeros recorded)
                       std::string> entry_type;   // 'revamp' return

  // DISABLED

private:

  LinkLogger(const LinkLogger& orig);             // copy constructor
  LinkLogger& operator= (const LinkLogger& orig); // copy assignment operator

  // CREATORS

public:

  LinkLogger();
  ~LinkLogger();

  // ACCESSORS

  std::string                                     // newline terminated
  recover                                         // dump database
  (const int tab = 4) const;                      // left indent

  int                                             // number of failed entries
  getNullCount() const;

  int                                             // number of successful entries
  getLogCount() const;

  // MANIPULATORS

  void
  reset();                                        // clear database, reestablish header

  bool                                            // simply return 'okay' value
  insert
  (const xeona::assign_ptr<Entity>& link,         // link entity
   const bool                       okay);        // return from 'revamp' call

  // INSTANCE DATA

private:

  unsigned                  d_linkNulls;          // number of null inserts
  std::vector<entry_type>   d_linkLog;            // database of live inserts

};

#endif // _LINKLOG_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : recset.h
//  file-create-date : Tue 09-Oct-2007 17:13 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : records and fields and also record-sets / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/recset.h $

//  HEADER GUARD

#ifndef _RECSET_H_
#define _RECSET_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/xeona_ptr.h"   // remappable counted pointer which mimics shared_ptr
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../a/exapp.h"       // application exception classes
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include "../b/entity.h"      // entity base class

#include <ostream>            // output streams
#include <sstream>            // string-streams
#include <stdexcept>          // standard exception classes, runtime_error()
#include <typeinfo>           // run-time type information (RTTI)
#include <vector>             // STL sequence container

#include <boost/variant.hpp>            // safe specified type heterogeneous storage
#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

using xeona::assign_ptr;                // remappable counted pointer

//  FORWARD (PARTIAL) DECLARATIONS

namespace xeona { extern bool nopro; }       // for function 'wrap<T>::wrapExit()'

//  CODE

// ---------------------------------------------------------
//  ENUM            : xeona::RecordKind
// ---------------------------------------------------------
//  Description  : set of possible record kinds
//  Role         : used during file parsing and writing to control parsing logic
// ---------------------------------------------------------
```

```
namespace xeona
{
  enum RecordKind
    {
      e_notRecord = 0,   // default is 0 anyway
      e_note,            // verbatim multi-line note
      e_program,         // program configuration information
      e_entity,          // model-specific entities
      e_end              // end of model marker (not exactly a 'record' as such)
    };

} // namespace xeona

// ------------------------------------------------------
//  ENUM            : xeona::FieldKind
// ------------------------------------------------------
//  Description  : set of possible field kinds
//  Role         : used during file parsing to control parsing logic
// ------------------------------------------------------

namespace xeona
{
  enum FieldKind
    {
      e_notField = 0,
      e_comment,         // comment line (cf remark within the field left side)
      e_input,           // externally supplied data
      e_output           // internally generated data
    };

} // namespace xeona

// ------------------------------------------------------
//  CLASS           : wrap <>
// ------------------------------------------------------
//  Description  : wraps a "single" so that it becomes a "non-temporary"
//  Role         : initializing reference data members in entity objects
//  Techniques   : operator T&, one form of the type-conversion operator
//  Status       : complete
//
//  Design notes
//
//      This utility class is used here to "wrap" a type which
//      the compiler has identified as a "temporary" and thereby
//      make it "acceptable" for binding to a reference:
//
//          int init = 10;      // 'init' seen as temporary by the compiler
//          wrap<int> w(init);  // create a class 'wrap' instance
//          // ...
//          int& r = w;   // bind 'w' to non-const reference 'r'
//          int  s = w;   // straight copy assignment, no cast necessary
//          w++;          // r,w = 11, s = 10, init = 10
//          r++;          // r,w = 12, s = 10, init = 10
//          s--;          // r,w = 12, s =  9, init = 10
//
//      The following statements are also usable:
//
//          wrap<int> copy1 = wrap<int>(3);
//          wrap<int> copy2 = w;
//          wrap<int> copy3 = static_cast<wrap<int> >(r);
//          wrap<int> copy4(w);
//
//      This class has the desired effect and no known
//      side-effects -- beyond making the type specifications
//      more complicated than they would otherwise be (for
//      instance, some functions return 'wrap<T>' rather than
//      straight 'T').
//
//      The idea draws on code in Lischner (2003 p119).  Please
//      study example 5-21 ("Binding references to conversion
//      lvalues") for further insights.
//
//      Note that the 'type-conversion operator' is not the same
//      as the 'address-of operator', although the signatures can
//      look quite similar.  The type-conversion operator has no
//      return type, not even 'void' or 'void*'.
//
//  CAUTION: default constructed object usage
//
//      Do NOT try to assign from a zero-argument (default)
//      constructed wrap object.  If you do, you should get a
```

```
//      short warning message explaining the problem before the
//      program seg-faults or exits() (depending on the code).
//
//  CAUTION: fully defined in header
//
//      This class is fully defined in the header file to avoid
//      template instantiation issues.
//
//  CAUTION: forward declaration
//
//      The correct way to forward declare this class is:
//
//          template <typename T> class wrap;
//
// ------------------------------------------------------------

template <typename T>
class wrap                                  // just for the type-conversion operator T&
{
public:
  explicit wrap() : d_value() { }           // mostly to member initialize 'd_single'
  explicit wrap(T v) : d_value(new T(v)) { }
  operator T& ()
  {
    if ( ! d_value ) wrapExit();            // else seg-faults on the return statement

    // reporting
    static logga::spLogger logger = logga::ptrLogStream();
    logger->repx(logga::adhc, "wrap return dereferenced d_value", *d_value);

    return *d_value;                        // return aliasable value
  }
private:
  void wrapExit()                           // warning message and exit
    throw(std::exception,                   // exception specification
          xeona::empty_wrap);
  shared_ptr<T>    d_value;                 // pointer usage is possibly essential [1]
};

// [1] pointer usage: things appear to work okay if 'd_value' is
// a straight T, but then I didn't test all the copy construction
// and copy assignment permutations.  That said, the shared_ptr
// design just felt like the more robust option in these other
// circumstances.  Moreover, the pointer approach default
// constructs an unusable object (an empty smart pointer),
// whereas the full type approach default constructs a 'T' with a
// usable but spurious value (often zero).

template <typename T>
void
wrap<T>::wrapExit()
  throw(std::exception,                     // exception specification
        xeona::empty_wrap)
{
  static logga::spLogger logger = logga::ptrLogStream();
  logger->repx(logga::dbug, "entering member function", "");
  const std::string type = xeona::demangle(typeid(T).name());
  if ( xeona::nopro == false )              // meaning option '--krazy' not applied
    {
      logger->repx(logga::xtra, "will throw xeona::empty_wrap", "");
      throw xeona::empty_wrap(type);
    }
  else
    {
      logger->repx(logga::dbug, "continuing under --krazy, type", type);
    }
}

// ------------------------------------------------------------
//  TYPEDEF         : single_type
//  TYPEDEF         : timeseries_type
// ------------------------------------------------------------
//  Description  : enable multiple value types to be stored together
//  Role         : used to hold record data also mirrored with entities
//  Techniques   : Boost.Variant library
//
//  Design notes
//
//      As described elsewhere, a "single" value has one element,
//      whereas a "timeseries" value has a "horizon" of elements.
//      In terms of algebra, these would map roughly to 'scalar'
```

```
//      and 'vector'.
//
//      The "single_type" requires the 'wrap' class template so
//      that it can be used to initialize reference data members
//      in entity objects.  See class 'wrap' for more details.
//
// ------------------------------------------------------

typedef boost::variant<
  wrap<int>,
  wrap<double>,
  wrap<bool>,
  wrap<tribool>,
  wrap<std::string>
> single_type;

typedef boost::variant<
  shared_ptr<std::vector<int> >,
  shared_ptr<std::vector<double> >,
  shared_ptr<std::vector<bool> >,
  shared_ptr<std::vector<tribool> >,
  shared_ptr<std::vector<std::string> >
> timeseries_type;

// ------------------------------------------------------
//  CLASS           : Field
// ------------------------------------------------------
//  Description  : holds field information, broken into various parts
//  Role         : used by Record objects
// ------------------------------------------------------

#ifdef _XUTEST                                  // normally defined when unit testing
namespace unittest { template <typename T> void unlexTestSingle(const T input); }
#endif // _XUTEST

class Field
{
  // FRIENDS

  friend class Record;                          // for access to private utility functions

#ifdef _XUTEST
  friend int main(int, char**);                 // unit test access to private functions
  template <typename T> friend void unittest::unlexTestSingle(const T); // CAUTION: [1]
#endif // _XUTEST

  // [1] function template friendship: note unusual syntax, for
  // which Lischner (2003 pp188-190) provides a full description.

  //  DISABLED

private:

  Field(const Field& orig);                     // copy constructor
  Field& operator= (const Field& orig);         // copy assignment operator

  // CREATORS

public:

  Field();

  // MANIPULATORS -- for loading data

public:

  void addName(std::string fieldname);
  void addKind(xeona::FieldKind kind);
  void addEnabled(tribool enabled);
  void addUnits(std::string units);
  void addRemark(const std::string& remark);
  void addRawStr(const std::string& rawStr);
  void addSplitStr(const std::vector<std::string>& splitStr);

  template <typename T> void addSingle(wrap<T>& single);
  template <typename T> void addTimeseries(shared_ptr<std::vector<T> > tseries);

// ACCESSORS

public:
```

```
  std::string                       getName()     const;     // d_name
  xeona::FieldKind                  getKind()     const;     // d_kind
  tribool                           getEnabled()  const;     // d_enabled
  std::string                       getUnits()    const;     // d_units
  std::string                       getRemark()   const;     // d_remark

  std::string                       getRawStr()   const;     // d_rawStr
  const std::vector<std::string>&   getSplitStr() const;     // d_splitStr (const ref)
  int                               getCount()    const;     // d_splitStr.size()
  bool                              isEmpty()     const;     // d_splitStr.empty()

  // ACCESSORS -- after binding has occurred

  const std::string              getSingle();
  const std::vector<std::string> getTimeseries()
    throw(xeona::empty_field_on_write);      // exception specification
  const std::string              getTimeseries(const std::string& sep);

  // MANIPULATORS -- for use by friendly Record objects

private:

  void
  splitRawStr()                             // process into split string
    throw(std::exception,                   // exception specification
          xeona::short_timeseries);

  template <typename T>
  void
  lexValue                                  // convert from string to nominated type
  (shared_ptr<std::vector<T> > output);

  void
  unlexSingle                               // convert back to split string vector
  (const single_type& input);               // CAUTION: & is necessary

  void
  unlexTimeseries                           // convert back to split string vector
  (const timeseries_type input);

  // INTERNAL DATA

private:

  std::string                d_name;        // name of field
  xeona::FieldKind           d_kind;        // either input, output, or comment
  tribool                    d_enabled;     // indeterminate indicates incomplete
  std::string                d_units;       // physical units, empty indicates dim'less
  std::string                d_remark;      // in-line remark
  std::string                d_rawStr;      // value as raw string (not used)
  std::vector<std::string>   d_splitStr;    // value as split string

  // safe storage of specified types using the Boost.Variant library

  single_type                d_single;
  timeseries_type            d_timeseries;

  // STATIC DATA

private:

  static logga::spLogger     s_logger;      // shared_ptr to single logger object

}; // class 'Field'

// HEADER-SIDE IMPLEMENTATIONS

template <typename T>
void
Field::addSingle
(wrap<T>& single)
{
  d_single = single;
}

template <typename T>
void
Field::addTimeseries
(shared_ptr<std::vector<T> > tseries)
{
  d_timeseries = tseries;
```

```
}

// ----------------------------------------------------------
//  CLASS           : Record
// ----------------------------------------------------------
//  Description  : holds a record, comprising metainfo and fields
//  Role         : used by RecordSet
// ----------------------------------------------------------

class Record
{
  // FRIENDS

  friend class RecordSet;                    // for access to private utility functions

#ifdef _XUTEST
  friend class Entity_UT;                    // for 'recset.ut0.cc'
#endif // _XUTEST

  // DISABLED

private:

  Record(const Record& orig);                // copy constructor
  Record& operator= (const Record& orig);    // copy assignment operator

  // CREATORS

public:

  Record();

  ~Record();

  // MANIPULATORS -- for loading data

public:

  void addIdentifier(std::string recordId);
  void addKind(xeona::RecordKind recordKind);
  void addEnabled(tribool recordEnabled);
  void addComment(const std::string& comment);
  void addField(shared_ptr<Field> field)
    throw(std::exception, xeona::xem_data_issue);      // exception specification

  // ACCESSORS

  const std::string                       getIdentifier() const;      // d_identifier
  xeona::RecordKind                       getKind()       const;      // d_kind
  tribool                                 getEnabled()    const;      // d_enabled

  const std::vector<std::string>&         getComments()   const;      // d_comments
  const std::vector<shared_ptr<Field> >&  getFields()     const;      // d_fields

  const std::string                       locateClass()   const;

  // BINDING FUNCTIONS -- for use by entities

public:

  template <typename T>                      // T must be supported by single_type
  wrap<T>                                    // see design notes for class 'wrap'
  tieSingle
  (const std::string& fieldname);

  template <typename T>                      // T in {int,double,bool,tribool,std::string}
  wrap<T>                                    // see design notes for class 'wrap'
  tieConstant
  (const T constantValue);

  template <typename T>                      // T must be supported by timeseries_type
  shared_ptr<std::vector<T> >                // only the contained type need be specified
  tieTimeseries
  (const std::string& fieldname)
    throw(xeona::timeseries_not_found);      // exception specification

  template <typename E>                      // CAUTION: defined in header file
  assign_ptr<E>
  lazyLink                                   // gets id, calls 'Entity' factory function
  (const std::string& linkname)
    throw (xeona::lazy_link_fail)
```

```
  {
    // locate the required field
    shared_ptr<Field> f;                    // field of interest
    f = locateField(linkname);              // called function warns if not found
    if ( ! f )
      {
        s_logger->repx(logga::warn, "field not found", linkname);
        if ( xeona::nopro == false )        // meaning option '--krazy' not applied
          {
            s_logger->repx(logga::warn, "will throw xeona::lazy_link_fail", "");
            throw xeona::lazy_link_fail("link name not found", linkname);
          }
      }

    // experience shows the both the lazy linking and the write
    // out fails if the string is not properly lexed and stored
    // -- hence the following is identical to the 'tieSingle'
    // code with T = std::string ALBEIT with a different return
    // type.

    // first, lex 'splitStr' as 'std::string'
    shared_ptr<std::vector<std::string> > lexed(new std::vector<std::string>());
    f->lexValue<std::string>(lexed);        // load temp with type T values

    // then, create an aliasable single as in 'tieSingle'
    std::string value = lexed->front();     // grab the first and only element
    wrap<std::string> single(value);        // create "wrapped" form
    f->addSingle<std::string>(single);      // copy assign by reference to the recordset

    // do not return 'single' here as we need a entity (smart)
    // pointer and not a T reference!

    const std::string id = value;           // CAUTION: 'getRawStr()' returns quoted form

    // reporting
    s_logger->repx(logga::dbug, "linkname (key)", linkname);
    s_logger->repx(logga::dbug, "recovered id (value)", id);

    // CAUTION: the 'makeLinkEntity' call needs "entity.h" 'hash-include'

    // finally invoke the factory function and return
    return Entity::makeLinkEntity<E>(id);

  } // member function 'lazyLink<>'

  // MANIPULATORS -- for use by friendly RecordSet objects

private:

#ifdef _XUTEST                              // .. and also for use by unit testd
public:                                     // CAUTION: access specifier in 'hash-ifdef'
#endif // _XUTEST

  const shared_ptr<Field>                   // returns empty shared_ptr if not found
  locateField
  (const std::string& fieldname) const;

  const shared_ptr<Field>                   // returns empty shared_ptr if not found
  locateFieldQuietly                        // non-logging version of above
  (const std::string& fieldname) const;

  void
  processFields();                          // cycles thru splitRawStr() calls

public:

  // MANIPULATORS -- for unit tests which make entities but do not read from a model file

  void
  hackUnitTestRecord                        // CAUTION: hollow function unless '_XUTEST'
  (const std::string builtinRemark,
   const int         steps = 2);            // note default

  // INTERNAL DATA

private:

  std::string                       d_identifier; // unique (tested later)
  xeona::RecordKind                 d_kind;       // enumeration
  tribool                           d_enabled;    // indeterminate indicates incomplete
  std::vector<shared_ptr<Field> >   d_fields;     // principal data
```

```
  std::vector<std::string>          d_comments;   // vector supports multi-line comments

  // STATIC DATA

private:

  static logga::spLogger            s_logger;     // shared_ptr to single logger object

}; // class 'Record'

// ----------------------------------------------------------
//  CLASS           : RecordSet
// ----------------------------------------------------------
//  Description  : holds a set of Records
//  Role         : provide a single point of entry
// ----------------------------------------------------------

class RecordSet
{
  // LOCAL ENUMERATIONS

public:

  enum Status
    {
      e_empty        = 0,                    // as constructed
      e_loaded       = 1,                    // the field value string is raw
      e_processed    = 2,                    // the field value string is processed
      e_bound        = 3,                    // the fields are typed and bound
      e_written      = 4                     // the records have been written out
    };

  // CREATORS

public:

  RecordSet();

  ~RecordSet();

  // MANIPULATORS -- for loading data and such

  void setStatus(Status status);
  void addRecord(shared_ptr<Record> record)
    throw(std::exception, xeona::xem_data_issue);      // exception specification
  void processRecords();

  // BOUND SUBSETS

  std::vector<shared_ptr<Record> >           // may be an empty vector
  copySet();

  std::vector<shared_ptr<Record> >           // may be an empty vector
  copySubset
  (const xeona::RecordKind recKind);

  // ACCESSORS

  Status                                  getStatus();
  unsigned                                getCount();
  const std::vector<shared_ptr<Record> >& getRecords();

  const shared_ptr<Record>
  locateRecord
  (const std::string& recordIdentifier) const;

  const shared_ptr<Record>
  locateRecordQuietly
  (const std::string& recordIdentifier) const;

  const bool
  existsEnabledRecord
  (const std::string& recordIdentifier) const;

  const shared_ptr<Field>
  locateRecordAndField
  (const std::string& recordIdentifier,
   const std::string& fieldname) const;

#ifdef _XUTEST // unit testing only
```

```
  void
  dump
  (std::string marker = "---");              // optional marker to insert

#endif // _XUTEST

  // INTERNAL DATA

private:

  Status                                d_status;
  std::vector<shared_ptr<Record> >      d_records;

  // STATIC DATA

private:

  static logga::spLogger                s_logger;     // shared_ptr to single logger object

}; // class 'RecordSet'

#endif // _RECSET_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : si3units.h
//  file-create-date : Thu 17-Dec-2009 13:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : support for engineering format and SI prefixes / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/si3units.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Heldoorn (2002) provides an excellent review of the SI
//  system, particularly in relation to usage and typesetting.
//
//    Heldoorn, Marcel.  2002.  The SIunits package : support
//      for the International System of Units [pdf file:
//      SIunits.pdf]

//  HEADER GUARD

#ifndef _SI3UNITS_H_
#define _SI3UNITS_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

  // ----------------------------------------------------------
  //  ENUM            : xeona::SiPrefix
  // ----------------------------------------------------------

namespace xeona
{
  enum SiPrefix
    {
      tooSmall     =  -1,                  // nonsensical value
      tooLarge     =  +1,                  // nonsensical value

      notSet       =  +2,                  // nonsensical value

      yocto        = -24,                  // y
      zepto        = -21,                  // z
      atto         = -18,                  // a
      femto        = -15,                  // f
      pico         = -12,                  // p
      nano         =  -9,                  // n
      micro        =  -6,                  // u (more properly \mu)
```

```
      milli        =  -3,                    // m
      none         =   0,                    //   (a space is currently used)
      kilo         =  +3,                    // k
      mega         =  +6,                    // M
      giga         =  +9,                    // G
      tera         = +12,                    // T
      peta         = +15,                    // P
      exe          = +18,                    // E
      zetta        = +21,                    // Z
      yotta        = +24                     // Y
    };

} // namespace 'xeona'

namespace xeona
{
  // ------------------------------------------------------
  //  FREE FUNCTION   : xeona::getSiPrefixStr
  // ------------------------------------------------------
  //  Description  : transform 'xeona::SiPrefix' to string
  //  Role         : general
  //  Techniques   : (nothing special)
  //  Status       : complete
  // ------------------------------------------------------

  std::string
  getSiPrefixStr
  (xeona::SiPrefix prefix);

  // ------------------------------------------------------
  //  FREE FUNCTION   : xeona::getEngineeringPrefix
  // ------------------------------------------------------
  //  Description  : obtain engineering prefix for 'value'
  //  Role         : general, for instance 8000 returns 'xeona::kilo'
  //  Techniques   : (nothing special)
  //  Status       : complete
  // ------------------------------------------------------

  xeona::SiPrefix
  getEngineeringPrefix
  (const double quantity,
   const int    shift = 0);                 // 1 means accept 4 digit numbers

  // ------------------------------------------------------
  //  FREE FUNCTION   : xeona::fmtEngineering
  // ------------------------------------------------------
  //  Description  : see example
  //  Role         : general
  //  Techniques   : relies on 'xeona::fmtEngineeringFix'
  //  Status       : complete
  //
  //  Examples
  //
  //      value  : 0.0000007700
  //      final  : 770.00 n
  //
  //      value  : -7.7
  //      final  : -7.70        (with two trailing spaces)
  //
  //      value  : -7777.7
  //      final  : -7.79 k
  //
  //      value  : -77777777777777.7
  //      final  : -77.79 T
  //
  // ------------------------------------------------------

  std::string
  fmtEngineering
  (const double   quantity,                // measured value
   const unsigned decimals = 2);           // number of decimals places

  // ------------------------------------------------------
  //  FREE FUNCTION   : xeona::fmtEngineeringFix
  // ------------------------------------------------------
  //  Description  : as above, but use given 'xeona::SiPrefix'
  //  Role         : general
  //  Techniques   : (nothing special)
  //  Status       : complete
  //
  //  Design notes
```

```
  //
  //      Read code regarding exact behavior.
  //
  // ----------------------------------------------------------

  std::string
  fmtEngineeringFix
  (double                quantity,           // measured value
   const xeona::SiPrefix level,              // such as 'xeona::kilo'
   const unsigned        decimals    = 2,    // number of decimal places
   const double          closeToZero = 1.0e-10);

} // namespace 'xeona'

#endif // _SI3UNITS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : rundev.h
//  file-create-date : Wed 25-Jul-2007 07:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : principal simulation call / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/simcall.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Principal simulation call.

//  HEADER GUARD

#ifndef _SIMCALL_H_
#define _SIMCALL_H_

//  LOCAL AND SYSTEM INCLUDES

#include <string>             // C++ strings

//  CODE

namespace xeona
{

  // ----------------------------------------------------------
  //  ENUM            : xeona::SimKind
  // ----------------------------------------------------------
  //  Description  : describe the kind of simulation to undertake
  //  Role         : argument to 'simulate'
  // ----------------------------------------------------------

  enum SimKind
  {
    e_notSpecified       = 0,
    e_hollowCall         = 1,
    e_identifyFile       = 2,
    e_parseModel         = 3,
    e_invokeFactory      = 4,
    e_linkAndConnect     = 5,
    e_firstStepRun       = 6,
    e_fullRun            = 7
  };

  // ----------------------------------------------------------
  //  ENUM            : xeona::SimRet
  // ----------------------------------------------------------
  //  Description  : encode the 'simulate' exist status
  //  Role         : end-of-application reporting
  // ----------------------------------------------------------
```

```
  enum SimRet
  {
    e_statusNotKnown     = 0,               // indeterminate state
    e_success            = 1,
    e_modelFileFault     = 2,
    e_infeasibility      = 3,               // thru solver failing or proven badness
    e_errantSimulation   = 4,
    e_testCodeUsed       = 8,               // meaning 'hash-ifdef' code and similar used
    e_other              = 9
  };

} // namespace 'xeona'

// ----------------------------------------------------------
//  FREE FUNCTION   : simulate
// ----------------------------------------------------------

xeona::SimRet                               // simulation return
simulate
(const std::string&   modelName,            // from command-line or default
 const xeona::SimKind simKind);             // from '--mode' option or default

#endif // _SIMCALL_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : stats.h
//  file-create-date : Wed 16-Sep-2009 13:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : on-the-fly statistical calculations / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/stats.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _STATS_H_
#define _STATS_H_

//  AD-HOC NOTES
//
//  Code originally developed in 'frag-stats-on-the-fly-3.cc'.

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <algorithm>          // STL copying, searching, and sorting
#include <cmath>              // C-style maths, ceil(), floor(), sqrt()
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ------------------------------------------------------------
//  documentation   : statistical calculations
// ------------------------------------------------------------
//
//  Source
//
//      Lischner (2003 pp269-270) with two corrections and some
//      additions.
//
//      Lischner notes: "pass an instance of 'Statistics' to the
//      'for_each' algorithm to accumulate the statistics. The
//      copy that is returned from 'for_each' contains the
//      desired results."  (p269)
//
//  Purpose
//
//      Compute statistics on the fly using a functor approach.
//      No copies of the original data are kept.
```

```
//
//  CAUTION: confirm formulas
//
//      Users should verify the statistical formulas for
//      themselves, particularly the (n - 1) divisor for
//      variance.
//
//  Further comments
//
//      The functor is templated, thereby allowing the calling
//      code to specify the type (fundamental or user-defined)
//      which most suits their requirements.
//
//      Non-computable expressions naturally return 'nan' (for
//      not-a-number), rather than throwing and then crashing.
//      Hence, empty and one-element datasets both yield
//      intuitive output.  This feature is rather convenient.
//
//  Tested
//
//      OUTPUT dataset is from Stephens etal (2006 p405).
//
//  Development environment
//
//      - GNU GCC 4.1.2 compiler (using -Wall -Weffc++ -pedantic)
//      - Linux Ubuntu 6.10 operating system (now superseded by 7.04)
//      - run-time memory checking with valgrind ver "valgrind-3.2.0-Debian"
//      - laptop is Toshiba Tecra A2 330, new 2004, 1.4GHz Intel Celeron M
//
//  Note on 'std::for_each'
//
//      'std::for_each' requires an operation (function or
//      functor) that uses call-by-reference semantics -- this
//      then allows elements to be modified directly (by
//      contrast, 'std::transform' uses call-by-value semantics
//      and requires a return value).  Josuttis (1999 p325)
//      discusses this topic in detail.
//
//  References
//
//    Josuttis, Nicolai M.  1999.  The C++ Standard Library : a
//      tutorial and reference.  Addison-Wesley, Boston, USA.  ISBN
//      0-201-37926-0.
//
//    Lischner, Ray.  2003.  C++ in a nutshell : a language and
//      library reference, O'Reilly and Associates, Sebastopol,
//      California, USA.  ISBN 0-596-00298-X.
//
//    Stephens, D Ryan, Christopher Diggins, Jonathan Turkanis,
//      and Jeff Cogswell.  2006.  C++ cookbook : solutions and
//      examples for C++ programmers.  O'Reilly Media, Sebastopol,
//      California, USA.  ISBN 0-596-00761-2.
//
// ------------------------------------------------------------

// ------------------------------------------------------------
//  CLASS           : Statistics <>
// ------------------------------------------------------------
//  Description  : calculate and store statistics on-the-fly
//  Role         : general use, including by 'xeona::fillStatistics'
//  Techniques   : header implementation
//  Status       : complete
//
//  CAUTION: (n - 1) divisor
//
//      Users should verify the statistical formulas for
//      themselves, particularly the (n - 1) divisor for
//      variance.
//
// ------------------------------------------------------------

template <typename T>
class Statistics
{
public:

  // CREATORS

  Statistics() :
    d_count(0),
    d_last(std::numeric_limits<T>::quiet_NaN()),  // set to nonsensical value
```

```
    d_sum(0),
    d_sumsq(0),
    d_min(+std::numeric_limits<T>::infinity()),   // set to maximum possible
    d_max(-std::numeric_limits<T>::infinity()),   // set to minimum possible
    d_zeros(0)
  {
  }

  // FUNCTION CALL OPERATOR

  // the 'void' function call operator is essentially wrapper to
  // the same operator carrying an 'inf' -- it is provided for
  // notational convenience

  void operator() ()
  {
    const T inf = std::numeric_limits<T>::infinity();
    operator()(inf);
  }

  void operator() (T x)
  {
    d_count += 1;
    d_last   = x;
    d_sum   += x;
    d_sumsq += x * x;
    if ( x < d_min ) d_min = x;
    if ( x > d_max ) d_max = x;
    if ( x == 0 ) d_zeros += 1;
  }

  // ACCESSORS

  operator bool() const { return ( d_count > 0 ); }         // 'false' is not yet loaded

  int    count()      const { return d_count; }
  T      last()       const { return d_last; }               // last entry
  T      sum()        const { return d_sum; }
  T      sumsq()      const { return d_sumsq; }
  T      mean()       const { return d_sum / d_count; }
  T      var()        const { return (d_sumsq - d_sum * d_sum / d_count) / (d_count - 1);}
  T      var2()       const { return (d_sumsq - d_sum * d_sum / d_count) / (d_count); }
  T      sdev()       const { return std::sqrt(var()); }     // refer <cmath>
  T      sdev2()      const { return std::sqrt(var2()); }
  T      min()        const { return d_min; }
  T      max()        const { return d_max; }
  T      range()      const { return d_max - d_min; }
  int    zeros()      const { return d_zeros; }
  int    nonzeros()   const { return d_count - d_zeros; }
  double zerorat()    const { return double(d_zeros)           / double(d_count); }
  double nonzerorat() const { return double(d_count - d_zeros) / double(d_count); }

  // INSTANCE DATA

private:

  // on-the-fly registers

  int    d_count;                           // was std::size_t (two places)
  T      d_last;                            // last entry
  T      d_sum;
  T      d_sumsq;                           // sum of squares
  T      d_min;
  T      d_max;
  int    d_zeros;

}; // class template 'Statistics<>'

// ------------------------------------------------------
//  FREE FUNCTION   : xeona::fillStatistics <>
// ------------------------------------------------------
//  Description  : simplify filling of 'Statistics' objects
//  Role         : general used, including entity 'conclude' calls
//  Techniques   : 'std::for_each'
//  Status       : complete
// ------------------------------------------------------

namespace xeona
{

  template <typename T>
```

```
  const Statistics<T>
  fillStatistics
  (const std::vector<T>& timeseries)
  {
    Statistics<T> stat = std::for_each
      (timeseries.begin(),
       timeseries.end(),
       Statistics<T>());
    return stat;
  }

  template <typename T>
  const Statistics<T>
  fillStatistics
  (const shared_ptr<std::vector<T> >& timeseries)
  {
    Statistics<T> stat = std::for_each
      (timeseries->begin(),
       timeseries->end(),
       Statistics<T>());
    return stat;
  }

} // namespace 'xeona'

#endif // _STATS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : tsops.h
//  file-create-date : Wed 16-Sep-2009 16:57 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : overloaded operators for timeseries / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/tsops.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _TSOPS_H_
#define _TSOPS_H_

//  AD-HOC NOTES
//
//  Code originally developed in 'frag-arith-overload-1.cc'.

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

//  CODE

// ----------------------------------------------------------
//  notes           : non-member overloaded operators
// ----------------------------------------------------------
//
//  Operators implemented as non-member functions
//
//      Dattatri (2002 pp398-401) covers such operators.  He
//      observes:
//
//          "Any operator that does not require an l-value ans is
//          commutative is better implemented as a non-member
//          function (+, -, etc).  This allows the compiler to
//          apply a conversion in case of argument mismatch for
//          the first argument." (p396)
//
//      Dattatri, Kayshav.  2002.  C++ : effective-object
//        oriented software construction : concepts, principles,
//        industrial strategies and practices -- Second edition.
//        Prentice Hall PTR, Upper Saddle River, New Jersey, USA.
//        ISBN 0-13-086769-1.
//
// ----------------------------------------------------------
```

```
// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::vectorSum <>
// ----------------------------------------------------------
//  Description  : sums a numerical vector
//  Role         : entity support
//  Techniques   : 'for' loop and 'operator+='
//  Status       : complete
//
//  Usage
//
//      double sum = xeona::vectorSum(vec);
//
//  Note an alternative approach
//
//      const Statistics<double> stat = xeona::fillStatistics(vec);
//      double sum                    = stat.sum();
//
// ----------------------------------------------------------

namespace xeona
{
  template <typename T>
  T
  vectorSum
  (const shared_ptr<std::vector<T> >& one);
}

// ----------------------------------------------------------
//  FREE FUNCTION   : operator+ <>
//  FREE FUNCTION   : operator- <>
// ----------------------------------------------------------
//  Description  : operates on two numerical vector
//  Role         : entity support
//  Techniques   : 'for' loop and underlying operator
//  Status       : complete
// ----------------------------------------------------------

template <typename T>
shared_ptr<std::vector<T> >
operator+
(const shared_ptr<std::vector<T> >& one,
 const shared_ptr<std::vector<T> >& two);

template <typename T>
shared_ptr<std::vector<T> >
operator-
(const shared_ptr<std::vector<T> >& one,
 const shared_ptr<std::vector<T> >& two);

#endif // _TSOPS_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : utils.h
//  file-create-date : Thu 14-Jun-2007 15:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions which offer general utilities 1 / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/util1.h $

//  HEADER GUARD

#ifndef _UTIL1_H_
#define _UTIL1_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../c/smart_ptr.h"   // toggle between TR1 and Boost smart pointers

#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

#include <boost/any.hpp>                // type heterogeneous storage
#include <boost/filesystem.hpp>         // path objects, iterators, useful operations
#include <boost/format.hpp>             // printf style formatting
#include <boost/date_time/posix_time/posix_time_duration.hpp>   // limited header

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  FORWARD (PARTIAL) DECLARATIONS

namespace xeona { extern const unsigned consoleWidth; }     // from "common.h"

//  CODE

namespace xeona
{

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::getRandom (int, int)
  // ----------------------------------------------------------
  //  Description  : retrieve random ints in required range
  // ----------------------------------------------------------

  int                                   // retrieve a random number
```

```
getRandom
(int lower,                             // inclusive lower bound
 int upper);                            // inclusive upper bound

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::getRandom (double, double)
// ----------------------------------------------------------
//  Description  : retrieve random doubles in required range
// ----------------------------------------------------------

double                                  // retrieve a random number
getRandom
(double lower,                          // inclusive lower bound
 double upper);                         // inclusive upper bound

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::fmtQuantity
// ----------------------------------------------------------
//  Description  : return a quantity in resolved SI format
// ----------------------------------------------------------

std::string
fmtQuatity
(const int         value,
 const std::string baseUnit);

std::string
fmtQuantity
(const double      value,               // numerical value
 const std::string baseUnit);           // for instance, 'W' for Watts

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::fmtPriceRate
// ----------------------------------------------------------
//  Description  : return a price rate in readable 'kilo' format
// ----------------------------------------------------------

std::string
fmtPriceRate
(const int         priceRate,           // numerical value
 const std::string baseUnit);           // for instance, 'W' for Watts

std::string
fmtPriceRate
(const double      priceRate,           // numerical value
 const std::string baseUnit);           // for instance, 'W' for Watts

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::vectorRepeat <>
// ----------------------------------------------------------
//  Description  : create a vector from a repeating pattern
//  Techniques   : explicit template instantiations in source file
// ----------------------------------------------------------

template <typename T>                   // meaning a std::vector<T> container
bool                                    // returns false if pattern is empty
vectorRepeat
(const std::vector<T>& pattern,         // pattern vector, must not be empty
 std::vector<T>&       fullset);        // output vector must be correct length

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::vectorReport <>
// ----------------------------------------------------------
//  Description  : print a vector to a nominated ostream
//  Role         : for testing purposes
//  Techniques   : fully defined in header to avoid template instantiation problems
//  Status       : complete
//
//  Usage
//
//      std::vector<int> vec(10, 2)               // contains ten twos
//      xeona::vectorReport(vec);                 // note implicit instantiation
//      xeona::vectorReport(vec, "\n", std::clog);  // one per line to stderr
//
//  Design notes
//
//      Unlike many sequence container display functions, this
//      code does not insert a trailing separator.
//
//      See Breyman (2000 p59) for the 'while' idiom.  See
//      Stephens etal (2006 pp356-357) for stream state code.
```

```
//
//      One improvement might be to utilize the Boost.Io_state
//      (i/o stream-state saver) library
//
// ------------------------------------------------------------

template <typename T>                       // meaning a std::vector<T> container
void
vectorReport
(const std::vector<T>& vec,                 // CAUTION: const requires const_iterator
 const std::string&    sep = " ",           // can be "\n"
 std::ostream&         os  = std::clog)     // defaults to stdlog
{
  std::ios::fmtflags prior = os.flags();    // save stream state
  os << std::boolalpha;                     // report bool and tribool in words
  typename std::vector<T>::const_iterator pos = vec.begin();
  if ( pos != vec.end() )
    os << *pos++;
  while ( pos != vec.end() )
    os << sep << *pos++;
  os << std::endl;
  os.flags(prior);
}

// ------------------------------------------------------------
//  FREE FUNCTION   : makeNullShared_ptr <>
// ------------------------------------------------------------
//  Description  : makes a null shared pointer
//  Role         : as required
//  Techniques   : fully defined in header to avoid template instantiation problems
//  Status       : complete
//
//  Note explicit calls (useful for static data initialization)
//
//      empty : shared_ptr<T>::shared_ptr()
//      null  : shared_ptr<T>::shared_ptr(static_cast<T*>(0))
//
// ------------------------------------------------------------

template <typename T>
inline
shared_ptr<T>
makeNullShared_ptr()
{
  // CAUTION: g++ requires "typename" for templated version

  return typename shared_ptr<T>::shared_ptr(static_cast<T*>(0));
}

// ------------------------------------------------------------
//  FREE FUNCTION   : reportShared_ptr <>
// ------------------------------------------------------------
//  Description  : report on pointer status
//  Role         : testing
//  Techniques   : fully defined in header to avoid template instantiation problems
//  Status       : complete
//
//  Design notes
//                        use_count()  unique()  get()    bool test
//    smart pointer ..    ---------------------------------------
//      which is empty           0        false     0         false
//      holding null pointer     1        true      0         false
//      not copied               1        true      0x000     true
//      with copies            > 1        false     0x000     true
//
//  Output
//
//      empty null single multiple(n)
//
// ------------------------------------------------------------

template <typename T>
std::string
reportShared_ptr
(const shared_ptr<T>& sp)
{
  std::ostringstream oss;
  switch ( sp.use_count() )
    {
    case 0:
      oss << "empty";
```

```
      break;
    case 1:
      if ( sp.get() == 0 )
        oss << "null";
      else
        oss << "single";
      break;
    default:
      oss << "multiple(" << sp.use_count() << ")";
      break;
    }
  return oss.str();
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::formatDuration
// ----------------------------------------------------------

std::string
formatDuration
(const boost::posix_time::time_duration& delta);

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::logDirect <>
// ----------------------------------------------------------
//  Description  : approximately mimics a 'Logger::repx' call
//  Role         : in circumstances where a normal call might be problematic
//  Techniques   : fully defined in header to avoid template instantiation problems,
//                 preprocessor macros
//  Status       : complete
//
//  Typical usage
//
//      xeona::logDirect(__FILE__, __LINE__, __func__, "destructor call", 1.23456789);
//
//    gives:
//
//      |  84   frag-log-direct-1.cc  main                --              NONE [slash]
//      |   destructor call                       1.23
//
// ----------------------------------------------------------

template <typename T>
inline
bool
logDirect
(const std::string file,                  // from preprocessor macro __FILE__
 const int         line,                  // from preprocessor macro __LINE__
 const std::string func,                  // from preprocessor macro __func__
 const std::string text,                  // message
 const T&          value,                 // comment/value
 std::ostream&     os = std::clog)        // reporting stream with default value
{
  if ( ! os )                             // this code assumes std::clog is always open
    {
      std::string msg = "nominated ostream not open, see";
      logDirect(__FILE__, __LINE__, __func__, msg, line + "  " + file);
      return false;
    }

  const std::string rank = "FUNC";        // 4 chars like a 'logga::Rank' string
  const unsigned TERM    = xeona::consoleWidth;      // set in "common.h"

  std::string macs;                       // duly formatted preprocessor macros
  std::string info;                       // text and value information
  std::string cons;                       // actual line of console output

  std::ostringstream ssValue;
  ssValue << std::fixed                   // floats only: never use scientific format
          << std::setprecision(2);        // .. and always print 2 decimal places
  ssValue << value;                       // utilize std::ostream conversions

  macs = boost::str(boost::format("%4s   %-15s  %s")    % line % file % func   );
  info = boost::str(boost::format("%-36s %s")           % text % ssValue.str() );
  cons = boost::str(boost::format("%-48s -- %17s   %s") % macs % rank % info   );

  if ( cons.length() > TERM )             // truncate if necessary
    {
      cons.resize(TERM - 2);
      cons += " >";                       // add truncation symbol
    }
```

```
  os << cons << "\n";                     // stream the collated information

  return true;
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::reverseSequence <>
// ----------------------------------------------------------
//  Description  : reverse and return a sequence container by value
//  Role         : general, tested using 'std::vector' 'std::deque' 'std::list'
//  Techniques   : fully defined in header to avoid template instantiation problems
//  Headers      : <algorithm>, <iterator>
//  Status       : complete
//
//  Design notes
//
//      Simple function to reverse the order in sequence
//      containers and return an 'rvalue'.
//
//      Implicit template instantiation is sufficient in many (if
//      not all) circumstances.
//
//      This call can be used in 'BOOST_FOREACH' constructs to
//      traverse backwards (unless your compiler is too old):
//
//          BOOST_FOREACH( shared_ptr<FullEntity> fe,
//                         xeona::reverseSequence(d_FullEntitys) )
//            {
//              fe->doSomething();
//            }
//
//      The 'reverse_copy' function template from <algorithm> is
//      used with the 'back_inserter' function template from
//      <iterator>.  This application is demonstrated in Josuttis
//      (1999 p364) with the comment "use back_inserter to insert
//      instead of overwrite".  For more information on the
//      'back_inserter' see Lischner (2003 p538) and Stephens
//      etal (2006 pp266-267).
//
//  CAUTION: supported sequence containers
//
//      tested using     : vector, deque, list
//      also works       : string (holding chars) (pseudo container)
//      not tested using : valarray (pseudo container)
//
// ----------------------------------------------------------

template <typename S>                     // 'S' is a specialized sequence container
S                                         // return an 'rvalue' of type 'S'
reverseSequence
(const S& forward)
{
  S reverse;
  std::reverse_copy(forward.begin(),                // refer <algorithm>
                    forward.end(),
                    std::back_inserter(reverse));   // refer <iterator>
  return reverse;
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::tailCombine <>
// ----------------------------------------------------------
//  Description  : append sequence container 'tail' to 'combine' and thus modify
//  Role         : general, specifically used to combine record subsets
//  Techniques   : explicit template instantiations in source file
//  Status       : complete
// ----------------------------------------------------------

template <typename S>                     // 'S' is a specialized sequence container
void
tailCombine
(S&       combined,                       // resultant container, need not be empty
 const S& tail);                          // container to be "back inserted"

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::mapCombine <>
// ----------------------------------------------------------
//  Description  : merge map 'addition' with map 'combined' and thus modify the latter
//  Role         : general, specifically written for registering gateways
//  Techniques   : explicit template instantiations in source file
```

```
  //  Caution      : requires UNIQUE keys, warnings issued for duplicates
  //  Status       : complete
  // ------------------------------------------------------------

  template <typename S>
  void
  mapCombine
  (S&       combined,
   const S& addition);

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::trimLeadingDigits
  // ------------------------------------------------------------
  //  Description  : trims leading digits from given string
  //  Role         : various, including use by 'xeona::demangle'
  //  Techniques   : C-style 'std::isdigit' from <cctype>
  //  Status       : complete
  // ------------------------------------------------------------

  std::string                              // this return may be streamed
  trimLeadingDigits
  (std::string str);                       // CAUTION: do not change to pass-by-ref

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::demangle
  // ------------------------------------------------------------
  //  Description  : attempts to reinterpret straightforward g++ typenames
  //  Role         : repairing typeid names from GCC RTTI
  //  Techniques   : simple reverse engineering
  //  Status       : complete
  // ------------------------------------------------------------

  std::string                              // this return may be streamed
  demangle                                 // GCC-specific for class typenames
  (std::string tid);                       // most often 'tid' is a 'const char*'

} // namespace xeona

#endif // _UTIL1_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : util2.h
//  file-create-date : Thu 06-Nov-2008 16:53 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions which offer general utilities 2 / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/util2.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _UTIL2_H_
#define _UTIL2_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <deque>              // STL sequence container, double-ended vector
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <sstream>            // string-streams
#include <stdexcept>          // standard exception classes, runtime_error()
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <cmath>              // C-style maths, abs(), ceil(), floor(), sqrt()

#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  CODE

namespace xeona
{

  // ----------------------------------------------------------
  //  FREE FUNCTION   : geometricProgression
  // ----------------------------------------------------------
  //  Description  : transform positive integer to geometric progression
  //  Role         : called by 'xeona::isTwoContained'
  //  Status       : complete
  //
  //  Examples
  //
  //           0 -> empty vector
```

```
  //           1 -> { 1 }
  //          30 -> { 0  2  4  8 16 }
  //          32 -> { 0  0  0  0  0 32 }
  //
  // ----------------------------------------------------------

  std::vector<int>
  geometricProgression
  (const int aggregate);

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::isTwoContained
  // ----------------------------------------------------------
  //  Description  : test if 'candidate' is a power-of-two sub-sequence of 'aggregate'
  //  Role         : used for checking commitment strategy codes
  //  On success   : returns 'true' if 'candidate' is contained in 'aggregate'
  //  On fail      : returns 'indeterminate' for faulty input
  //  Status       : complete
  //
  //  Design notes
  //
  //      This routine uses conventional arithmetic, but it can
  //      also be done with bitwise operators (if I am not
  //      mistaken).
  //
  //      Note 'pure' must be a power-of-two greater than zero.
  //      But 'aggregate' may be less than 'pure' and can be zero
  //
  // ----------------------------------------------------------

  tribool                                   // 'indeterminate' means faulty input
  isTwoContained
  (const int candidate,                     // greater than zero
   const int aggregate);                    // non-negative, can be less than 'candidate'

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::reducedVector (string return)
  // ----------------------------------------------------------
  //  Description  : returns string
  //  Role         :
  //  Techniques   : (nothing special)
  //  Status       : incomplete
  // ----------------------------------------------------------

  std::string                               // no trailing newline
  reducedVector
  (const int         aggregate,
   const std::string separator = " ");

} // namespace 'xeona'

// ----------------------------------------------------------
//  CLASS           : SmartPtrPopper <>
// ----------------------------------------------------------
//  Description  : small popable container class
//  Role         : to assist 'capsolve' algorithm
//  Techniques   : 'std::deque', header implementation to simplify template instantiation
//  Status       : complete
// ----------------------------------------------------------

template <typename E>
class SmartPtrPopper
{
  // DISABLED

private:

  SmartPtrPopper(const SmartPtrPopper& orig);               // copy constructor
  SmartPtrPopper& operator= (const SmartPtrPopper& orig);   // copy assignment operator

  // CREATORS

public:

  SmartPtrPopper();

  // ACCESSORS

  int
  size();
```

```
  bool                                  // similar to 'std::vector::empty'
  empty();

  // MANIPULATORS

  int
  load
  (std::vector<shared_ptr<E> > spvec);

  int
  push                                  // similar to a returning 'push_front'
  (shared_ptr<E> sp);

  shared_ptr<E>                         // returns empty pointer if deque is empty
  pop();                                // similar to a returning 'pop_front'

private:

  std::deque<shared_ptr<E> >    d_spdeq;

};

// the 'SmartPtrPopper' class is implemented in the header in
// order to avoid explicit template instantiations

// CREATORS

template <typename E>
SmartPtrPopper<E>::SmartPtrPopper() :
  d_spdeq()
{
}

// ACCESSORS

template <typename E>
int
SmartPtrPopper<E>::size()
{
  return d_spdeq.size();
}

template <typename E>
bool
SmartPtrPopper<E>::empty()
{
  return d_spdeq.empty();
}

// MANIPULATORS

template <typename E>
int
SmartPtrPopper<E>::load
(std::vector<shared_ptr<E> > spvec)
{
  d_spdeq.clear();                      // empty all elements
  std::copy(spvec.begin(),              //  refer <algorithm>, note 'reverse_copy'
            spvec.end(),
            std::back_inserter(d_spdeq));    // refer <iterator>
  return d_spdeq.size();
}

template <typename E>
int
SmartPtrPopper<E>::push
(shared_ptr<E> sp)
{
  d_spdeq.push_front(sp);
  return size();
}

template <typename E>
shared_ptr<E>
SmartPtrPopper<E>::pop()
{
  if ( d_spdeq.empty() )
    {
      return shared_ptr<E>();           // empty shared pointer
    }
  else
```

```
    {
      shared_ptr<E> temp = d_spdeq.front();  // grab first element
      d_spdeq.pop_front();                   // remove first element
      return temp;
    }
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::coeffProduct <>
// ------------------------------------------------------------
//  Description  : calculate an inner product while honoring constant term
//  Role         : calculating objective results without re-calling the solver
//  Techniques   : 'std::inner_product'
//  Status       : complete
//
//  Equation
//
//      answer = a_0 + sum( a_i * x_i )   for i in { 1, .., n }
//
//          with x_0 ignored for calculation purposes
//          and the vectors a and x of length n+1
//
// ------------------------------------------------------------

namespace xeona
{
  template<typename N>                      // implicit instantiation supported
  N
  coeffProduct
  (const std::vector<N> a,                  // coefficients vector
   const std::vector<N> x);                 // variable values vector

} // namespace 'xeona'

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::isNan <>
// ------------------------------------------------------------
//  Description  : checks for IEEE 754 NaN
//  Role         : general usage
//  Techniques   : based on string-streaming (not classy), implicit template instantiation
//  Status       : complete
//
//  Design notes
//
//      NaN is floating point not-a-number.  Here are some ways
//      to make NaNs:
//
//          0.0/0.0
//          std::sqrt(-1.0)                           // <cmath>
//          std::numeric_limits<double>::quiet_NaN()  // refer <limits>
//
//      NaN's are equal to nothing, not even themselves.
//      Moreover NaN's have no sign.
//
//  Usage
//
//      if ( xeona::isNan(value) )  // NaN
//      else                        // not-a-NaN
//
// ------------------------------------------------------------

namespace xeona
{
  template<typename T>
  bool
  isNan
  (const T number)
    throw(std::domain_error)                // exception specification
  {
    // abort if 'T' is an integral type
    if ( std::numeric_limits<T>::is_integer )
      {
        std::ostringstream oss;
        oss << "caller passed a non-floating point type of value: " << number;
        throw std::domain_error(oss.str());
      }
    // main code
    std::ostringstream oss;                 // refer <sstream>
    oss << number;                          // stream 'number'
    if ( oss.str() == "nan" ) return true;
    else                      return false;
```

```
  }
} // function 'xeona::isNan'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::isInf <>
// ----------------------------------------------------------
//  Description  : checks for IEEE 754 inf
//  Role         : general usage
//  Techniques   : implicit template instantiation
//  Status       : complete
//
//  Design notes
//
//      inf is floating point infinity.
//
//      Unlike NaN's, inf's can be compared and posses a sign.
//
//  Usage
//
//      if ( xeona::isInf(value) )  // inf or -inf
//      else                        // not-an-inf
//
// ----------------------------------------------------------

namespace xeona
{
  template<typename T>
  bool
  isInf
  (const T number)
    throw(std::domain_error)                // exception specification
  {
    // abort if 'T' is an integral type
    if ( std::numeric_limits<T>::is_integer )
      {
        std::ostringstream oss;
        oss << "caller passed a non-floating point type of value: " << number;
        throw std::domain_error(oss.str());
      }
    // main code
    if ( std::abs(number) == std::numeric_limits<double>::infinity() ) return true;
    else                                                               return false;
  }
} // function 'xeona::isInf'

#endif // _UTIL2_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : util3.h
//  file-create-date : Thu 31-Dec-2009 00:17 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions for floating point comparison / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/util3.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Optimization '-O2' and better produces compile warnings under
//  '-Wall'.  This relates to code in free function
//  'xeona::almostEqual' and more particularly to the lines:
//
//          long long aInt = *(long long*)&A;
//          long long bInt = *(long long*)&B;
//
//  CAUTION: note that 'long long' is a g++ extension.  If other
//  compilers are deployed then this code will need to be somehow
//  reworked.
//
//  This is a SERIOUS warning and CANNOT be ignored.
//
//  More explicitly:
//
//      warn options  : -Wall
//      optimizations : -O3
//      macros        : -D_XTCOLS=142 -D_XSVNREV=4215 -D_XRELEASE -DNDEBUG
//
//      c/util3.cc: In function
//        'bool xeona::almostEqual(double, double, long long int)':
//      c/util3.cc:332 (and 333): warning:
//        dereferencing type-punned pointer will break strict-aliasing rules
//
//  The fix was to add the following undocumented, but widely
//  discussed, g++ compiler option:
//
//      -fno-strict-aliasing
//
//  In more depth:
//
//      This issue is tripped under '-Wstrict-aliasing' which is
//      included under '-Wall' AND when optimization option
//      '-fstrict-aliasing' is in force, as bundled under '-O2'
//      or better.
//
//  Finer grained fixes might include:
//
//      Attribute 'may_alias':
//
//           this might work, see
//           file:///usr/share/doc/gcc-4.1-base/gcc.html
```

```
//
//           typedef long long __attribute__((__may_alias__)) mylonglong;
//
//      Use of 'pragma':
//
//          There may also be GCC mechanisms for turning off
//          optimizations for individual functions.  First note
//          that GCC function attributes don't apply in this case
//          and that these pragma features might only apply to
//          Ada and not C/C++.
//
//      Keyword 'volatile':
//
//          The keyword 'volatile' might help, whilst noting that
//          this directive is probably only ADVISORY and
//          therefore leading, if it does test out, to brittle
//          code.

//  HEADER GUARD

#ifndef _UTIL3_H_
#define _UTIL3_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <sstream>            // string-streams
#include <stdexcept>          // standard exception classes, runtime_error()
#include <string>             // C++ strings

//  COMPILE-TIME ASSERTIONS

#ifndef __GNUG__ // not GNU g++ compiler
#  warning "this unit contains g++ compiler-specific code (namely use of type long long)"
#endif // __GNUG__

//  CODE

// ------------------------------------------------------------
//  ENUM            : xeona::Precision
// ------------------------------------------------------------
//  Description  : enum to describe levels of "almost equal" precision
//  Role         : used in function 'xeona::almostEqual' (enum-based wrapper)
//  Status       : complete
// ------------------------------------------------------------

namespace xeona
{
  enum Precision
    {
      exact     = 1,                        // same floating point representation
      numic     = 2,                        // numerically "equal"
      tight     = 3,                        // tightly "equal"
      loose     = 4                         // loosely "equal"
    };

} // namespace 'xeona'

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::almostEqual (enum-based wrapper)
// ------------------------------------------------------------
//  Description  : compare two floats approximately using enum-based thresholds
//  Role         : general use, with support for predefined levels of precision
//  Techniques   : overloading, enum 'xeona::Precision'
//  Status       : complete
//
//  Design notes
//
//      First see also the workhorse version documentation.
//
//      NOTE: the percentages are only accurate for given example
//      -- they can vary within a factor of two for other
//      comparisons as discussed elsewhere.
//
//      reference : 3.333333333
//      exact     : only 3.333333333 passes           nil
//      numic     : 3.33333  passes  3.3333  fails    +/- 0.0001%
//      tight     : 3.333    passes  3.33    fails    +/- 0.01%
//      loose     : 3.3      passes  3.0     fails    +/- 10%
```

```
//
//  Usage as a conditional
//
//      if ( xeona::almostEqual(A, B, xeona::tight) )
//        {
//          // equal to within about 0.01%
//        }
//
// ----------------------------------------------------------

namespace xeona
{
  bool
  almostEqual
  (const double           A,
   const double           B,
   const xeona::Precision precision);        // maps to 'maxUlps' (see below)
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::almostEqual (workhorse)
// ----------------------------------------------------------
//  Description  : compare two floats approximately using max-ulps value
//  Role         : general use, but with more control than  wrapper version
//  Techniques   : overloading, 'xeona::isInfinite', C-style casting, integer arithmetic
//  Status       : complete
//
//  Overview
//
//      First see also the implementation file documentation.
//
//      The 'maxUlps' argument is the number of distinct floats
//      lying between 'A' and 'B' -- if zero, this call is then
//      equivalent to ( A == B ) but very much slower.
//
//      This comparison is neither equivalent to absolute error
//      nor relative error.  But it does map quite closely to
//      relative error, albeit with a two-fold variability
//      depending on where specifically the floats lie in terms
//      of their exponent range.
//
//      (With apologies to Orwell, some floats are more equal!)
//
//  Usage as a conditional
//
//      if ( xeona::almostEqual(A, B, 10000000000LL) )
//        {
//          // equal to within the specified number
//          // of intervening floats
//        }
//
//      The integer literal should be (case-insensitive)
//      postfixed "LL" as shown above.
//
//      As coded, this function does not successfully equate
//      'inf' and big numbers, for example:
//
//          double dblmax = std::numeric_limits<double>::max();
//          double inf    = std::numeric_limits<double>::infinity();
//          xeona::almostEqual(dblmax, inf, 10000000000LL); // false
//
//      See the unit test test-suites for more working examples.
//
//  Throws
//
//      This function will throw a 'std::domain_error' if passed
//      a negative 'maxUlps' value (the throw code could easily
//      be removed to create a no-throw version if desired).
//
// ----------------------------------------------------------

namespace xeona
{
  bool
  almostEqual
  (const double     A,
   const double     B,
   const long long  maxUlps)                // maximum units of last place
    throw(std::domain_error);               // exception specification

} // namespace 'xeona'
```

```
#endif // _UTIL3_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xemgen.h
//  file-create-date : Mon 11-Aug-2008 14:46 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : class to generate well-formatted XEM models / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/xemgen.h $

//  HEADER GUARD

#ifndef _XEMGEN_H_
#define _XEMGEN_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : XemGenerator
// ----------------------------------------------------------
//  Description  : generate well-formatted XEM model data
//  Role         : used by '--inbuilt' and various model testers
//  Techniques   : Boost.Format library
//  Status       : complete
//
//  CAUTION: string arguments
//
//      These should remain "const std::string" and not be "const
//      std::string&" -- otherwise string literals cannot be
//      placed in function calls.
//
// ----------------------------------------------------------

class XemGenerator
{
private:

  enum EntryKind
    {
      e_unknown    = 0,
      e_record,
      e_input,                                // with '>'
      e_output,                               // with '<'
      e_comment,
      e_rule,
```

```
      e_ident,                                  // $Key: $
      e_meta,                                   // xem-role:
      e_verbatim
    };

private:

  XemGenerator(const XemGenerator& orig);              // copy constructor
  XemGenerator& operator= (const XemGenerator& orig);  // copy assignment operator

public:

  XemGenerator                                  // CAUTION: cannot disable zero-argument ctor
  (const int tab = 45);                         // note default

  ~XemGenerator();

  // RECORD LEVEL CALLS

  void
  entity
  (const std::string klass,
   const std::string identifier,
   const std::string preamble = "entity");   // else "inbuilt"

  void
  special
  (const std::string special);

  void
  note();                                       // usually followed by comment calls

  // FIELD LEVEL CALLS

  void
  in
  (const std::string key,
   const std::string value);

  void
  out
  (const std::string key,
   const std::string value);

  void
  inq
  (const std::string key,
   const std::string quote);                // quote as single string

  void
  outq
  (const std::string key,
   const std::string quote);

  void
  inQ                                           // multi-quote variant
  (const std::string key,
   const std::string quote);                // quote as timeseries string

  void
  outQ                                          // multi-quote variant
  (const std::string key,
   const std::string quote);                // quote as timeseries string

  // PACKAGED CALLS

  void
  horizon
  (const int steps,
   const int interval = 3600);              // note default

  void
  overseer();

  void
  emacs();                                      // insert emacs settings in note block

  // OTHER CALLS

  void
  com();                                        // blank line with in a set of comment
```

```
  void
  com                                       // comment
  (const std::string comment,
   const unsigned    pad
   = xeona::modelFieldIndent + 2);          // defaults to standard comment padding

  void
  hed                                       // header: with correct indent
  (const std::string header,
   const unsigned    pad
   = xeona::modelFieldIndent);

  void
  ident                                     // "Revision" to "$Revision: 4766 $"
  (const std::string key,
   const unsigned    pad
   = xeona::modelFieldIndent);              // defaults to standard alignment

  void
  meta                                      // "role" to "xem-role:"
  (const std::string key,
   const unsigned    pad
   = xeona::modelFieldIndent);              // defaults to standard alignment

  void
  verbatim
  (const std::string verbatim);

  void
  rule
  (const std::string header);

  void
  end();                                    // end marker

  void
  blanks
  (const unsigned count);

  // OUTPUT

  void
  print
  (std::ostream& os);

  std::string
  string();

private:

  void
  field
  (std::string key,
   std::string angle,
   std::string value);

private:

  EntryKind                 d_last;         // previous entry kind
  const int                 d_tab;          // alignment tab for '<' and '>'
  std::ostringstream        d_oss;          // internal buffer

  static logga::spLogger    s_logger;       // shared_ptr to single logger object

};

#endif // _XEMGEN_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xeona_ptr.h
//  file-create-date : Fri 31-Jul-2009 16:26 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : remappable counted pointer which mimics shared_ptr / header
//  file-status      : first-pass complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/xeona_ptr.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This is "header-only" code -- meaning that 'xeona_ptr.cc' is
//  empty (and indeed only present to facilitate both object file
//  building and linking and also unit testing).
//
//  Unlike most other units, class template 'assign_ptr<>' does
//  NOT use 'xeona'-style logging, but rather relies on the
//  locally-defined 'Deport' class.  The default output stream
//  was (at the time of writing) set to 'std::clog'.
//
//  It is intended that this code could be used in a stand-alone
//  context with a minimum of change.  However, the hash-includes
//  and namespace declarations from 'smart_ptr' will need to be
//  expressly provided.  Furthermore, the <typeinfo> and
//  <stdexcept> code is protective and can removed without loss
//  of core functionality.

//  HEADER GUARD

#ifndef _XEONA_PTR_H_
#define _XEONA_PTR_H_

//  AD-HOC NOTES
//
//    design issues
//
//      - get the caller-side C type information working in 'reVamp'
//
//    implementation issues and/or bugs
//
//      - the data members in 'xeona::applied_ptr<>' should be private but are public
//      - resolve the 'remove_if' "problem"
//      - perhaps offer more control over 'Deport' verboseness

//  LOCAL AND SYSTEM INCLUDES

#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <functional>         // STL function objects
#include <iomanip>            // setw() and family
```

```
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <stdexcept>          // standard exception classes, runtime_error()
#include <string>             // C++ strings
#include <typeinfo>           // run-time type information (RTTI), std::bad_cast exception
#include <vector>             // STL sequence container

#include <boost/any.hpp>      // safe type-heterogeneous storage
#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro
#include <boost/format.hpp>   // printf style formatting

// ELECT USAGE AS INTERNAL OR EXTERNAL TO 'XEONA'

#define XE_ASSIGN_PTR_XEONA_USE              // active means usage by 'xeona'

// ---------------------------------
//  internal use code
// ---------------------------------

#ifdef XE_ASSIGN_PTR_XEONA_USE

#  include "../a/logger.h"    // run-time logging functionality (as required)

// verbosity control
// 0 = fully silent, 1 = warn messages, 2 = internal, 3 = + ctors, 4 = + dtors
#  ifdef _XUTEST                               // originally _XDEBUG
    namespace { const unsigned verbose = 4; }
#  else
    namespace { const unsigned verbose = 0; }
#  endif // _XUTEST

// ---------------------------------
//  external use code
// ---------------------------------

#else // XE_ASSIGN_PTR_XEONA_USE

  namespace xeona { const bool nopro        = false; }
  namespace xeona { const int  exit_success = 0; }
  namespace xeona { std::string demangle (std::string tid) { return tid; } }

#  include <boost/shared_ptr.hpp>                // Boost shared pointer (serializable)
#  include <boost/weak_ptr.hpp>                  // Boost weak pointer (serializable)
#  include <boost/enable_shared_from_this.hpp>   // Boost smart pointer support

  using boost::shared_ptr;
  using boost::weak_ptr;
  using boost::enable_shared_from_this;
  using boost::const_pointer_cast;
  using boost::static_pointer_cast;
  using boost::dynamic_pointer_cast;
  using boost::bad_weak_ptr;                     // exception class

#endif // XE_ASSIGN_PTR_XEONA_USE

//  CODE

// ----------------------------------------------------------
//  notes           : xeona::assign_ptr overview
// ----------------------------------------------------------
//  Description  : "remappable" counted pointer which partly mimics 'shared_ptr'
//  Role         : primarily developed for the 'xeona' lazy link process
//  Techniques   : pointer semantics, "revamp" function, 'shared_ptr'
//  Status       : first-pass complete
//
//  Design notes
//
//      This reference counted pointer class has the ability to
//      switch the controlled resource via the 'revamp' function
//      -- a feature which is intrinsically IMPOSSIBLE with
//      'shared_ptr', as described here:
//
//          Boost Libraries
//          + Smart Pointers
//            + shared_ptr class template
//              + Smart Pointer Programming Techniques
//                + Obtaining a shared_ptr from a raw pointer
//
//      Having said that, this class template relies on
//      'shared_ptr's to do most of its work!
//
```

```
//      The idea of self-implemented counted pointers came from
//      Dattatri (2002 pp487-496).
//
//      The interface and usage is modeled on 'shared_ptr'
//      although not all its free and member functions are
//      offered here.  Read the code for details.
//
//      No attempt has been made to make this code run fast.
//      Rather the emphasis has been on design cleanliness and
//      maintainability.
//
//      It is recommended practice not to deploy smart pointers
//      as unnamed temporaries.
//
//      See also:
//      '/usr/local/include/boost-1_39/boost/smart_ptr/shared_ptr.hpp'
//
//  Potential extensions
//
//      Add a 'subsume' function, perhaps by applying
//      'std::set_union' to 'd_clients':
//
//          void subsume(assign_ptr<T> inferiorPool)
//
//      Note that 'swap' and some other functions from
//      'shared_ptr' are not implemented, neither are some of the
//      associated free functions.
//
//      Neither is any support offered for interoperation with
//      'weak_ptr' and 'auto_ptr'.
//
//      The use of custom 'shared_ptr' deleters, passed thru to
//      the underlying resource, could also be considered.
//
//  CAUTION: avoid language-defined types
//
//      This class should not be used on language-defined types
//      such as 'int' because '->' produces strange semantics
//      (Dattatri 2002 p496).
//
//  CAUTION: heap objects only
//
//       Stack objects should not be deployed and will yield
//       run-time faults on attempted deletion.
//
//  CAUTION: 'shared_ptr' construction is non-exclusive
//
//      The use of pre-existing 'shared_ptr' objects in
//      construction and also in 'revamp' and 'reset' calls means
//      that the resource might not be exclusive.  This
//      eventuality can be tested using the member function
//      'exclusive'.
//
//  CAUTION: possibly thread-UNSAFE as implemented
//
//      Consult Dattatri for ideas as to how one might confirm or
//      make this class thread-safe.
//
//  Background
//
//      Alexandrescu (2001, ch7, pp157-195) devotes a chapter to
//      smart pointers and the underlying design trade-offs --
//      however "revamping" is not considered:
//
//      The notion of "revamping" is never far away in Meyers
//      (1996 item29 pp183-213) reference counting example, but
//      neither is the concept directly mentioned.
//
//      Some ideas on the use of the Boost.Any library came from
//      Karlsson (2006 ch6 pp159-189).
//
//      Finally, Dattatri (2002 pp487-496) provides an
//      implementation of counted pointers and discusses the
//      design issues in some depth.
//
//  References
//
//      Alexandrescu, Andrei.  2001.  Modern C++ design : generic
//        programming and design patterns applied.  Addison-Wesley,
//        Boston, USA.  ISBN 0-201-70431-5.
//
```

```
//      Dattatri, Kayshav.  2002.  C++ : effective-object
//        oriented software construction : concepts, principles,
//        industrial strategies and practices -- Second edition.
//        Prentice Hall PTR, Upper Saddle River, New Jersey, USA.
//        ISBN 0-13-086769-1.
//
//      Karlsson, Bjoern.  2006.  Beyond the C++ Standard Library
//        : an introduction to Boost.  Addison-Wesley, Upper Saddle
//        River, New Jersey, USA.  ISBN 0-321-13354-4.
//
//      Meyers, Scott.  1996.  More effective C++ : 35 new ways
//        to improve your programs and design.  Addison-Wesley,
//        Boston, USA.  ISBN 0-201-63371-X.
//
// ----------------------------------------------------------

// ----------------------------------------------------------
//  CLASS           : ::Deport (local)
// ----------------------------------------------------------
//  Description  : reporting class
//  Role         : debug reporting
//  Techniques   : file scope, Boost.Format library, '::verbose'
//  Status       : complete
//
//  Overall verbosity control via local constant '::verbose'
//
//      0 = fully silent
//      1 = warn (right-side) messages
//      2 = all internal messages
//      3 = above plus constructor reports
//      4 = above plus destructor reports
//
// ----------------------------------------------------------

namespace
{
  class Deport
  {
    // LOCAL ENUMERATIONS

  private:

    enum Verbose
      {
        silent = 0,
        report = 1
      };

    // DISABLED

  private:

    Deport();                                 // zero-argument constructor
    Deport(const Deport& orig);               // copy constructor
    Deport& operator= (const Deport& orig);  // copy assignment operator

  public:

    // CREATORS

    explicit
    Deport
    (const std::string& host,                     // class name
     const std::string& call,                     // member function name
     const Verbose      verbose = report,
     std::ostream&      ostream = std::clog) :    // default ostream set here
       d_ostream(ostream),
       d_flag("-- assign pointer : "),            // margin flag string set here
       d_lead(boost::str(boost::format("%s %12s : %-28s") % d_flag % host % call)),
       d_verbose(verbose)
    {
      // assumes that the object instantiation call was placed at
      // the beginning of the function block
      if ( ::verbose > 2 ) operator()("enter", "", d_verbose);
    }

    ~Deport()
    {
      if ( ::verbose > 3 ) operator()("exit", "", d_verbose);
    }
```

```
    // MEMBER FUNCTIONS

    void
    operator()
    (const std::string& msg1,                // left column message
     const std::string& msg2  = "",          // right column message
     const Verbose      verbose = report)    // can set to 'Deport::silent'
    {
      if ( ::verbose == 0 )                  return;   // '::verbose' defined in file
      if ( ::verbose == 1 && msg2.empty() )  return;
      if ( verbose == silent)                return;   // 'verbose' is an argument
      if ( msg1.empty() && msg2.empty() )    return;

      d_ostream << d_lead << " : ";
      if ( msg2.empty() )
        d_ostream << msg1 << std::endl;
      else
        d_ostream << boost::format("%-30s  %38s **") % msg1 % msg2 << std::endl;
    }

    // INSTANCE DATA

  private:

    std::ostream&        d_ostream;
    const std::string    d_flag;
    const std::string    d_lead;
    Verbose              d_verbose;          // private enum

  };

} // unnamed namespace

namespace xeona
{

  // ------------------------------------------------------
  //  CLASS           : xeona::bad_assign_ptr
  // ------------------------------------------------------
  //  Description  : C++ exception class
  //  Role         : (none thus far)
  //  Techniques   : C++ exception mechanism
  //  Status       : complete
  // ------------------------------------------------------

  class bad_assign_ptr
    : public std::exception
  {
  public:

    bad_assign_ptr() :
      d_message()
    {
    }

    bad_assign_ptr
    (const std::string message) :
      d_message(message)
    {
    }

    ~bad_assign_ptr() throw()                // no-throw
    {
    }

    virtual
    char const*
    what() const throw()                     // no-throw
    {
      std::string msg = "xeona::bad_assign_ptr";
      if ( ! d_message.empty() ) msg += ": " + d_message;
      return msg.c_str();
    }

  private:

    const std::string    d_message;

  };

  // ------------------------------------------------------
```

```
  //  CLASS           : xeona::bad_assign_cast
  // ------------------------------------------------------------
  //  Description  : C++ exception class
  //  Role         : as required
  //  Techniques   : C++ exception mechanism
  //  Status       : complete
  // ------------------------------------------------------------

  class bad_assign_cast
    : public std::exception
  {
  public:

    bad_assign_cast() :
      d_message()
    {
    }

    bad_assign_cast
    (const std::string message) :
      d_message(message)
    {
    }

    ~bad_assign_cast() throw()               // no-throw
    {
    }

    virtual
    char const*
    what() const throw()                     // no-throw
    {
      std::string msg = "xeona::bad_assign_cast";
      if ( ! d_message.empty() ) msg += ": " + d_message;
      return msg.c_str();
    }

  private:

    const std::string    d_message;

  };

  // ------------------------------------------------------------
  //  CLASS           : xeona::const_cast_tag
  //  CLASS           : xeona::dynamic_cast_tag
  //  CLASS           : xeona::polymorphic_cast_tag
  // ------------------------------------------------------------
  //  Description  : built into overloaded cast constructors
  //  Role         : used by the associated free function
  //  Techniques   : 'struct'
  //  Status       : complete
  // ------------------------------------------------------------

  struct const_cast_tag       {};
//struct static_cast_tag      {};            // not currently implemented
  struct dynamic_cast_tag     {};
  struct polymorphic_cast_tag {};

  // ------------------------------------------------------------
  //  CLASS           : xeona::Client (abstract base class)
  // ------------------------------------------------------------
  //  Description  : interface class for class 'ClientImp<>'
  //  Role         : support client registration
  //  Techniques   : inheritance, pure virtual functions
  //  Status       : complete -- but see comments elsewhere about polymorphic behavior
  // ------------------------------------------------------------

  class Client
  {
    // DISABLED

  private:

    Client& operator= (const Client& orig);  // copy assignment operator

  public:

    // CREATORS

    virtual
```

```
  ~Client()
  {
    Deport deport("Client", "destructor");
  }

  Client()
  {
    Deport deport("Client", "constructor");
  }

  Client(const Client& orig)               // copy constructor, required by 'clone'
  {
  }

  // ACCESSORS

  virtual
  const std::type_info&
  type() const = 0;

  // MANIPULATORS

  virtual
  Client*                                  // derived class template is 'ClientImp<C>*'
  get() = 0;

  virtual
  bool
  update
  (shared_ptr<void> replacement) = 0;      // CAUTION: smart 'void*' generic pointer

  virtual
  bool
  updateAny
  (const boost::any& replacement) = 0;     // CAUTION: payload will need to carry a 'C'

};

// ----------------------------------------------------------
//  CLASS           : xeona::ClientImp <>
// ----------------------------------------------------------

template <typename T> class assign_ptr;    // forward (partial) declaration

template <typename C>
class ClientImp
  : public Client
{
  // DISABLED

private:

  ClientImp();                                          // zero-argument constructor
  ClientImp& operator= (const ClientImp& orig);         // copy assignment operator

public:

  ~ClientImp()
  {
    Deport deport("ClientImp", "destructor");
  }

  explicit
  ClientImp
  (assign_ptr<C>* client) :
    Client(),
    d_client(client)
  {
    Deport deport("ClientImp", "constructor");
    std::ostringstream oss;
    oss << "C = " << xeona::demangle(typeid(**client).name());
    deport(oss.str());
  }

  ClientImp(const ClientImp& orig) :       // copy constructor, required by 'clone'
    d_client(orig.d_client)                // shallow copy is correct
  {
  }

  // COMPARATORS
```

```
// logical equality function
friend
bool
operator== (const ClientImp<C>& a,
            const ClientImp<C>& b)
{
  Deport deport("free func", "ClientImp operator==");
  return a.d_client == b.d_client;
}

// ACCESSORS

virtual
const std::type_info&
type() const
{
  return typeid(C);
}

// MANIPULATORS

// "If the function's return type is a pointer (or a
// reference) to a base class, the derived class's function
// may return a pointer (or reference) to a class derived
// from that base class." (Meyers 1996 pp126-127).

virtual
ClientImp<C>*
get()
{
  return this;
}

virtual
bool
update
(shared_ptr<void> replacement)
{
  //  WARNING: the generic pointer signature here defeats the
  //  strong type system!  And in particular, the following
  //  is not caught because 'static_pointer_cast' and not
  //  'dynamic_pointer_cast' (see below) must be used:
  //
  //     assign_ptr<Next> next(new Next());
  //     next.revamp(new Base());
  //
  //  Shared_ptr type 'void' is deployed because the
  //  following declaration will not compile:
  //
  //      template <typename Y>
  //      virtual
  //      bool update(shared_ptr<Y> globalReplacement)

  // reporting
  Deport deport("ClientImp", __func__);

  // key call
  d_client->d_resource = static_pointer_cast<C>(replacement);

  // check result and return accordingly
  if ( d_client->d_resource == 0 )
    {
      deport("cast from void*", "BAD STATIC_POINTER_CAST");
      return false;                       // return failure
    }
  else
    {
      return true;                        // return success
    }
}

virtual
bool
updateAny
(const boost::any& replacement)
{
  Deport deport("ClientImp", __func__);

  // CAUTION: The Boost.Any cast specifier must be exact --
  // which means the 'replacement' payload must carry a 'C'
  // and nothing else.
```

```
    // CAUTION: the definition of 'buffer' in an 'if' test is
    // legal and remains visible in both the 'if' and 'else'
    // blocks (Alexandrescu 2001 p238).

    if ( const shared_ptr<C>* buffer = boost::any_cast<shared_ptr<C> >(&replacement) )
      {
        d_client->d_resource = *buffer;    // guaranteed type match on good cast
        deport("good boost::any_cast");
        return true;
      }
    else
      {
        deport("bad boost::any_cast");
        return false;
      }
  }

  template <typename Y>
  bool
  updateNonVirtual
  (shared_ptr<Y> replacement)
  {
    // reporting
    Deport deport("ClientImp", __func__);

    // key call
    d_client->d_resource = replacement;

    // return success
    return true;
  }

  // INSTANCE DATA

private:

  assign_ptr<C>*    d_client;

};

// ------------------------------------------------------------
//  CLASS           : xeona::Common
// ------------------------------------------------------------
//  Description  : helper class, holds client list, manages the revamp process
//  Role         : held by 'assign_ptr' as an aggregate data member
//  Techniques   : 'std::vector', casting
//  Status       : complete -- but could refactor the 'std::remove_if' predicate
//
//  Design notes
//
//  CAUTION: 'std::remove_if' design flaws
//
//      The member function 'detach' MAY NEED more work related
//      to the 'std::remove_if' predicate -- in particular read
//      Karlsson (2006 pp186-188), Josuttis (1999 pp302-304,
//      378-379), and Lischner (2003 p355).  Note that Lischner
//      mistakenly states that 'remove_if' acts on 'false'.
//
//      Indeed, the functor class template 'match' should
//      either be wrapped by or reimplemented as a predicate
//      function.  Moreover, the predicate function/functor
//      could also meet the STL library adapter requirements.
//
//      This is only a problem for 'std::remove_if'.
//
// ------------------------------------------------------------

template <typename T> class assign_ptr;    // forward (partial) declaration
template <typename M> class match;         // forward (partial) declaration

class Common
{
  // DISABLED

private:

  Common(const Common& orig);              // copy constructor
  Common& operator= (const Common& orig);  // copy assignment operator

public:
```

```
// CREATORS

~Common()                                // destructor
{
  Deport deport("Common", "destructor");
}

Common() :                               // zero-argument constructor
  d_clients(),                           // empty vector
  d_revampCount(0)
{
  Deport deport("Common", "constructor");
}

// GENERAL PURPOSE FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : xeona::Common::attach <>
// ----------------------------------------------------------

// register me
template <typename T>
void attach(assign_ptr<T>* client)
{
  Deport deport("Common", __func__);

  shared_ptr<Client> temp(new ClientImp<T>(client));
  d_clients.push_back(temp);

  std::ostringstream oss;
  oss << "client count = " << d_clients.size()
      << ", "
      << "revamp count = " << d_revampCount;
  deport(oss.str());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : xeona::Common::detach <>
// ----------------------------------------------------------

// unregister me
template <typename T, typename Y>
void detach(assign_ptr<Y>* client)
{
  Deport deport("Common", __func__);

  shared_ptr<ClientImp<Y> > temp(new ClientImp<Y>(client));

  // delete (remove and erase) ALL matches -- whilst noting
  // that 'erase' is a 'std::vector' member function and
  // 'std::remove_if' is from <algorithm>

  const unsigned before = d_clients.size();        // [1]
  d_clients.erase(std::remove_if(d_clients.begin(),
                                 d_clients.end(),
                                 match<Y>(temp)),  // [2]
                  d_clients.end());

  // [1] size type: strictly speaking, member function 'size'
  // returns type 'std::vector<shared_ptr<Client>
  // >::size_type' -- which is 'unsigned' on my system
  //
  // [2] CAUTION: 'std::remove_if' predicate: see notes
  // elsewhere!

  const int diff = before - d_clients.size();
  if ( diff != 1 )
    {
      deport("unique removal failed", "DIFF NOT ONE");
      if ( xeona::nopro == false )       // meaning option '--krazy' not applied
        {
          std::ostringstream ess;
          ess << "'xeona::assign_ptr<T>::reset' 'Common::detach'"
              << " coding error, 'diff' not one but " << diff;
          throw std::logic_error(ess.str());
        }
    }
  std::ostringstream oss;
  oss << "diff = " << diff;
  deport(oss.str());
```

```
    }

    // ------------------------------------------------------------
    //  MEMBER FUNCTION : xeona::Common::reVamp <>
    // ------------------------------------------------------------
    //
    //  CAUTION: incomplete design
    //
    //      This code does not implement resource convertibility
    //      in a way that covers all cases and is type-checked --
    //      in fact, achieving both would appear to be a major
    //      challenge.
    //
    //      Check the pre-processor macros in this code to see
    //      which conversions are currently supported.
    //
    //  Type-convertibility requirements
    //
    //          type T - results from the caller of 'revamp'
    //          type C - relates to the CURRENT client under traversal
    //          type Y - is from the replacement object
    //
    //      type-convertibility (where "convertible" implies "is
    //      or derives from" -- in other words, only direct
    //      assignment or implicit upcasting are acceptable):
    //
    //          Y convertible to C convertible to T
    //
    //      T and Y are known directly from this call under
    //      compile-time polymorphism
    //
    //      C is known indirectly thru run-time polymorphism and
    //      the 'Client' class hierarchy, namely:
    //
    //          class Client
    //          template <typename C> class ClientImp : public Client
    //
    //      Alexandrescu (2001 p264) gives an overview of
    //      polymorphism in C++.
    //
    // ------------------------------------------------------------

    template <typename T, typename Y>        // 'T' is nominated in call, 'Y' is implicit
    bool reVamp(shared_ptr<Y> globalReplacement)
    {
      Deport deport("Common", __func__);

      // administration
      d_revampCount++;
#ifdef XE_ASSIGN_PTR_XEONA_USE               // indicates 'xeona' proper
      bool retval = false;                   // assume false initially
#else
      bool retval = true;                    // assume true initially
#endif // XE_ASSIGN_PTR_XEONA_USE
      // set client count
      std::ostringstream oss1;
      oss1 << "client count = " << d_clients.size();
      const std::string clients = oss1.str();;

      // initial reporting
      deport(clients);

      // loop the clients
      int loopCount = 0;
      BOOST_FOREACH( shared_ptr<Client> client, d_clients )
        {
          // update and set loop count
          ++loopCount;
          std::ostringstream oss;
          oss << "count = " << std::setw(2) << std::setfill(' ') << loopCount;
          const std::string loops = oss.str();

          // reporting
          deport("loop enter, " + loops);

          // integrity check
          if ( client == 0 )
            {
              deport("client equals zero", "EMPTY OR NULL CLIENT");
              continue;
            }
```

```
#ifdef XE_ASSIGN_PTR_XEONA_USE              // indicates 'xeona' proper

          // ---------------------------------
          //  'xeona' code
          // ---------------------------------

          // CAUTION: special code for 'xeona' in which 'client'
          // must be exactly of type 'Y' for an update to occur
          // -- this code uses Boost.Any -- see r3280 for earlier
          // code based on typeid comparison and 'void*'

          const boost::any payload = globalReplacement;               // naturally Y
          const bool retY          = client->updateAny(payload);      // [1] key call

          // [1] this call uses "boost::any<shared_ptr<C> >" as
          // its cast specifier, which means that C and Y must
          // match for a successful outcome.

          static logga::spLogger logger = logga::ptrLogStream();      // bind logger
          static int loggerCount = 0;                                 // just for interest

          if ( retY )
            {
              // only one such update per pool should occur
              retval = true;                  // one or more true means success
              deport("client and resource type match", "UPDATE");
              logger->repx(logga::dbug, "update occured", ++loggerCount);
            }
          else
            {
              // CAUTION: at one stage, 'client->d_resource' was
              // set to empty pointer "shared_ptr<T>()" but this
              // created a bad bug whereby the 'xeona' 'polylink'
              // caller object was disabled and then destroyed
              // before the call had completed and returned.

              deport("client and resource type differ - skipping");
              logger->repx(logga::adhc, "update skipped", "");
            }

#else // XE_ASSIGN_PTR_XEONA_USE, in other words, code external to xeona

          // ---------------------------------
          //  normal code
          // ---------------------------------

          std::ostringstream oss8;
          oss8 << "payload = " <<  xeona::demangle(client->type().name());
          deport(oss8.str());

# if 0 // 0 = type-unsafe code, 1 = process only Y- and T-payloads

          // try Y payload
          const boost::any payloadY = globalReplacement;
          if ( client->updateAny(payloadY) == false )
            {
              // try T payload
              const boost::any payloadT = dynamic_pointer_cast<T>(globalReplacement);
              if ( client->updateAny(payloadT) == false )
                {
                  retval = false;             // one or more false means failure
                }
            }

# else

          if ( client->update(globalReplacement) == false )      // key call
            {
              retval = false;                 // one or false false means failure
            }

# endif // 0

          // BUGGY: direct calls not working as expected
          //
          // client->get()->updateNonVirtual(globalReplacement);
          // client->clone()->updateNonVirtual(globalReplacement);
          //
          // note: error: 'class xeona::Client' has
          // no member named 'updateNonVirtual'
```

```
          // ---------------------------------

#endif // XE_ASSIGN_PTR_XEONA_USE

          deport("loop leave");

        } // BOOST_FOREACH

      std::ostringstream oss2;
      oss2 << "revamp count = " << d_revampCount;
      const std::string revamps = oss2.str();
      deport(clients + ", " + revamps);

      // return statement
      return retval;
    }

    long clientCount() const
    {
      Deport deport("Common", __func__);
      return d_clients.size();
    }

    long revampCount() const
    {
      Deport deport("Common", __func__);
      return d_revampCount;
    }

    // INSTANCE DATA

  private:

    std::vector<shared_ptr<Client> >    d_clients;          // actually 'ClientImp<C>'
    long                                d_revampCount;      // initially zero

  };

  // ------------------------------------------------------------
  //  CLASS (FUNCTOR) : xeona::match <>
  // ------------------------------------------------------------

  template <typename M>
  class match
  {
    // DISABLED

  private:

    match();                                    // zero-argument constructor
    match& operator= (const match& orig);      // copy assignment operator

    // CREATORS

  public:

    match
    (const shared_ptr<ClientImp<M> >& target) :
      d_target(target),
      d_match(0),
      d_count(new int(0))
    {
      Deport deport("match", "constructor", Deport::silent);
      deport("in constructor");
      if ( d_target == 0 )
        {
          deport("target equals zero", "BAD TARGET");
        }
    }

    match(const match& orig) :                 // copy constructor
      d_target(orig.d_target),                  // shallow copy is correct
      d_match(orig.d_match),
      d_count(orig.d_count)                     // shallow copy is correct
    { }

    ~match()                                    // destructor
    {
      Deport deport("match", "destructor", Deport::silent);
      deport(boost::str(boost::format("matches = %d, count = %d") % d_match % *d_count));
```

```
  }

  // FUNCTION CALL OPERATOR

  bool
  operator()
  (const shared_ptr<Client>& current)
  {
    Deport deport("match", "function call operator", Deport::silent);
    std::stringstream oss;
    oss << "client = " << current
        << ", "
        << "target = " << d_target;
    deport(oss.str());

    shared_ptr<ClientImp<M> > downcast = dynamic_pointer_cast<ClientImp<M> >(current);
    if ( downcast == 0 )
      {
        deport("Client downcast failed", "BAD DOWNCAST");
        return false;
      }

    const bool match = ( *downcast == *d_target );   // object-wise comparison
    if ( match == true )
      {
        ++d_match;
        ++*d_count;
      }
    return match;
  }

  // CAUTION: [1] according to Josuttis (1999 p378) 'remove_if'
  // acts on 'true' but Lischner (2003 p355) says 'remove_if'
  // acts on 'false' -- however Josuttis is correct.

  // INSTANCE DATA

private:

  const shared_ptr<ClientImp<M> >    d_target;
  int                                d_match;
  shared_ptr<int>                    d_count;

};

// ------------------------------------------------------------
//  CLASS           : xeona::assign_ptr <>
// ------------------------------------------------------------
//  Description  : "remappable" counted pointer which partly mimics 'shared_ptr
//  Role         : point of access
//  Techniques   : 'shared_ptr'
//  Status       : complete -- but see preliminary notes for potential extensions
//
//  Template arguments
//
//      'T' is invoked class
//      'Y' is argument class, which needs to be a sub-class of 'T'
//      'U' is argument class, which needs to be a super-class of 'T'
//
// ------------------------------------------------------------

template <typename T>
class assign_ptr
{
  // FRIENDS

  friend class Common;                     // used by Common::reVamp

public:

  // CREATORS

  // UPCASTABLE: the various Y-to-T creators (more precisely,
  // function templates) require that the underlying resource
  // be naturally convertible from 'shared_ptr<Y>' to
  // 'shared_ptr<T>'.

  virtual ~assign_ptr();                         // destructor

  explicit
  assign_ptr();                                  // zero-argument (hollow) constructor
```

```
template <typename Y>
explicit assign_ptr(Y* resource);             // upclass raw pointer constructor

// NON-EXCLUSIVITY: the provision of an overloaded
// 'shared_ptr' constructor together with the equivalent
// 'revamp' and 'reset' member functions require the client to
// acknowledge that non-exclusive use may result -- in other
// words, not all instances of the controlled resource are
// under the jurisdiction of the current 'assign_ptr' pool.

template <typename Y>                         // can be NON-EXCLUSIVE
explicit assign_ptr(shared_ptr<Y> resource);  // upclass shared pointer constructor

assign_ptr(const assign_ptr& orig);           // normal copy constructor

template <typename Y>
assign_ptr(const assign_ptr<Y>& orig);        // upclass copy constructor

// COPY ASSIGNMENT OPERATORS

assign_ptr<T>& operator= (const assign_ptr& orig);     // copy assignment operator

template <typename Y>
assign_ptr<T>& operator= (const assign_ptr<Y>& orig);  // upclass copy assignment oper

// POINTER SEMANTICS

operator bool() const;                    // object status test
T& operator*  () const;                   // dereference operator
T* operator-> () const                    // member access operator
  throw (std::logic_error);               // exception specification

// COMPARATORS -- FREE FUNCTIONS

// logical equality function
template <typename Y>
friend
bool
operator== (const assign_ptr&    a,
            const assign_ptr<Y>& b);

// logical NOT equality function
template <typename Y>
friend
bool
operator!= (const assign_ptr&    a,
            const assign_ptr<Y>& b);

// POOL-WIDE FUNCTIONS

template <typename Y>
bool revamp(Y* resource);                 // replace resource for entire pool

template <typename Y>                     // can be NON-EXCLUSIVE
bool revamp(shared_ptr<Y> resource);      // replace resource for entire pool

long client_count() const;                // pool count
bool unique() const;                      // returns 'true' if sole client

long revamp_count() const;                // revamp count
bool original() const;                    // returns 'true' if never revamped

long external_count() const;                   // number of "uncontrolled" resources
bool exclusive() const;                        // 'true' is no "uncontrolled" resource

// INDIVIDUAL-ACTION FUNCTIONS

template <typename Y>
void reset();                             // split off from existing pool

template <typename Y>
void reset(Y* resource);                  // split off from existing pool

template <typename Y>                     // can be NON-EXCLUSIVE
void reset(shared_ptr<Y> resource);       // split off from existing pool

// DOWNCAST CONSTRUCTORS - NORMALLY INTERFACED BY FREE FUNCTIONS

template <typename U>
assign_ptr(assign_ptr<U> const & pointer,     // 'U' is the "cast to" type
```

```
            xeona::const_cast_tag);

  template <typename U>
  assign_ptr(assign_ptr<U> const & pointer,     // 'U' is the "cast to" type
            xeona::dynamic_cast_tag);          // returns zero on cast failure

  template <typename U>
  assign_ptr(assign_ptr<U> const & pointer,     // 'U' is the "cast to" type
            xeona::polymorphic_cast_tag)
    throw(std::bad_cast);                      // throws on cast failure

  // REPORT FUNCTION

  std::string                                  // with trailing newline
  report(const std::string& identifier = "(not supplied)",
         const std::string& comment    = "",
         const unsigned     leftIndent = 2) const;

  // CODE DEVELOPMENT CALLS

  T*             get() const;                  // null not tested, unlike 'operator->'
  assign_ptr<T>* get_assign(); // const         // simply returns 'this'

  void trash() { revamp(static_cast<T>(0)); }   // DANGEROUS

  // INSTANCE DATA

public:  // [1]

  shared_ptr<T>         d_resource;        // locally held resource
  shared_ptr<Common>    d_common;

  // [1] CAUTION: data access: the upclass copy constructor
  // requires that these be public -- when they really should
  // be private (I remain puzzled!).

};

// ---------------------------------
//  implementation
// ---------------------------------

// DESTRUCTOR

// destructor
// no action required due to use of 'shared_ptr's
template <typename T>
assign_ptr<T>::~assign_ptr()
{
  Deport deport("assign_ptr", "destructor");
}

// CONSTRUCTORS

// zero-argument (hollow) constructor
// makes a new 'Common' in the process
template <typename T>
assign_ptr<T>::assign_ptr() :
  d_resource(),
  d_common(new Common())
{
  Deport deport("assign_ptr", "constructor - hollow");
  deport("using new Common()");
}

// upcast raw pointer constructor
template <typename T>
template <typename Y>
assign_ptr<T>::assign_ptr(Y* resource) :
  d_resource(shared_ptr<Y>(resource)),
  d_common(new Common())
{
  Deport deport("assign_ptr", "constructor - raw upcast");
  d_common->attach(this);
}

// upcast shared pointer constructor
template <typename T>
template <typename Y>
assign_ptr<T>::assign_ptr(shared_ptr<Y> resource) :
  d_resource(resource),
```

```
  d_common(new Common())
{
  Deport deport("assign_ptr", "constructor - shared upcast");
  d_common->attach(this);
}

// normal copy constructor
template <typename T>
assign_ptr<T>::assign_ptr(const assign_ptr& orig) :
  d_resource(orig.d_resource),
  d_common(orig.d_common)
{
  Deport deport("assign_ptr", "copy constr - no cast");
  d_common->attach(this);
}

// upcast copy constructor (requires public data members)
template <typename T>
template <typename Y>
assign_ptr<T>::assign_ptr(const assign_ptr<Y>& orig) :
  d_resource(orig.d_resource),
  d_common(orig.d_common)
{
  Deport deport("assign_ptr", "copy constr - upcast");
  d_common->attach(this);
}

// COPY ASSIGNMENT OPERATORS

// copy assignment operator
template <typename T>
assign_ptr<T>& assign_ptr<T>::operator= (const assign_ptr& orig)
{
  Deport deport("assign_ptr", "copy assign - no cast");
  this->d_resource = orig.d_resource;
  this->d_common   = orig.d_common;
  d_common->attach(this);
  return *this;
}

// upcast copy assignment operator
template <typename T>
template <typename Y>
assign_ptr<T>& assign_ptr<T>::operator= (const assign_ptr<Y>& orig)
{
  Deport deport("assign_ptr", "copy assign - upcast");
  if ( this != &orig )                    // protect against self-assignment
    {
      this->d_resource = orig.d_resource;
      this->d_common   = orig.d_common;
      d_common->attach(this);
    }
  return *this;
}

// OVERLOADED OPERATORS

// bool operator
//
// based on 'std::tr1::shared_ptr' code, see also Becker (2007
// pp35-36) for a discussion on this same operator for
// 'shared_ptr' --this streams as 1/0 or true/false, if used
// as follows:
//
//     std::cout << myAssignPtr << std::endl;
//
template <typename T>
assign_ptr<T>::operator bool() const
{
  return d_resource == 0 ? 0 : d_resource.get();
}

// dereference operator *
template <typename T>
T& assign_ptr<T>::operator* () const
{
  return *d_resource;
}

// member access operator ->
template <typename T>
```

```
T* assign_ptr<T>::operator-> () const
  throw (std::logic_error)                 // exception specification
{
  if ( d_resource.get() == 0 )
    {
      if ( xeona::nopro == false )         // meaning option '--krazy' not applied
        {
          const std::string emsg
            = "'xeona::assign_ptr<T>::operator->' call on null resource";
          throw std::logic_error(emsg);
        }
    }
  return d_resource.get();
}

// COMPARATORS -- FREE FUNCTIONS

// friends injected into enclosing namespace and found by
// argument dependent (name) lookup (ADL)

// logical equality function
template <typename T, typename Y>
bool
operator== (const assign_ptr<T>& a,
            const assign_ptr<Y>& b)
{
  return a.get() == b.get();
}

// logical NOT equality function
template <typename T, typename Y>
bool
operator!= (const assign_ptr<T>& a,
            const assign_ptr<Y>& b)
{
  return !(a == b);
}

// POOL-WIDE FUNCTIONS

// raw revamp
// wrapper to shared pointer version
template <typename T>
template <typename Y>
bool
assign_ptr<T>::revamp
(Y* resource)
{
  Deport deport("assign_ptr", "revamp - raw");
  shared_ptr<Y> namedTemporary(resource);
  return revamp(namedTemporary);
}

// shared revamp
template <typename T>
template <typename Y>
bool
assign_ptr<T>::revamp
(shared_ptr<Y> resource)
{
  Deport deport("assign_ptr", "revamp - shared");
  std::ostringstream oss;
  oss << "T = " << xeona::demangle(typeid(T).name())
      << "   "
      << "Y = " << xeona::demangle(typeid(Y).name());
  deport(oss.str());

  const bool ret = d_common->reVamp<T>(resource);
  return ret;
}

// client count
template <typename T>
long
assign_ptr<T>::client_count() const
{
  return d_common->clientCount();
}

// unique test
template <typename T>
```

```
bool
assign_ptr<T>::unique() const
{
  return ( client_count() == 1 );
}

// revamp count
template <typename T>
long
assign_ptr<T>::revamp_count() const
{
  return d_common->revampCount();
}

// original test
template <typename T>
bool
assign_ptr<T>::original() const
{
  return ( revamp_count() == 0 );
}

// external count
template <typename T>
long
assign_ptr<T>::external_count() const
{
  return d_resource.use_count() - client_count();
}

// exclusivity test
template <typename T>
bool
assign_ptr<T>::exclusive() const
{
  return ( external_count() == 0 );
}

// INDIVIDUAL-ACTION FUNCTIONS

// empty reset
// wrapper to shared pointer version
template <typename T>
template <typename Y>
void
assign_ptr<T>::reset()
{
  Deport deport("assign_ptr", "reset - empty");
  reset(shared_ptr<Y>());                  // empty shared pointer
}

// raw reset
// wrapper to shared pointer version
template <typename T>
template <typename Y>
void
assign_ptr<T>::reset
(Y* resource)
{
  Deport deport("assign_ptr", "reset - raw");
  shared_ptr<Y> namedTemporary(resource);
  reset(namedTemporary);
}

// shared reset
template <typename T>
template <typename Y>
void
assign_ptr<T>::reset(shared_ptr<Y> resource)
{
  Deport deport("assign_ptr", "reset - shared");
  // CAUTION: note the explicit 'T' template argument below
  d_common->detach<T>(this);
  d_common = shared_ptr<Common>(new Common());
  d_common->attach<T>(this);
  d_resource = resource;
}

// DOWNCAST CONSTRUCTORS - NORMALLY INTERFACED BY FREE FUNCTIONS

template <typename T>
```

```
template <typename U>
assign_ptr<T>::assign_ptr                  // a kind of copy constructor
(assign_ptr<U> const & pointer,            // 'Y' is the "cast to" type
 xeona::dynamic_cast_tag) :                // tag 'struct' defined earlier
   d_resource(dynamic_pointer_cast<T>(pointer.d_resource)),
   d_common(pointer.d_common)
{
  Deport deport("assign_ptr", "dynamic cast ctor");
  if ( d_resource == 0 )                   // the cast failed
    {
      deport("resource equals zero", "CAST FAILED");
      d_common = shared_ptr<Common>(new Common());
    }
  d_common->attach(this);
}

template <typename T>
template <typename U>
assign_ptr<T>::assign_ptr                  // a kind of copy constructor
(assign_ptr<U> const & pointer,            // 'Y' is the "cast to" type
 xeona::const_cast_tag) :                  // tag 'struct' defined earlier
   d_resource(const_pointer_cast<T>(pointer.d_resource)),
   d_common(pointer.d_common)
{
  Deport deport("assign_ptr", "const cast ctor");
  // it is believed that this cast cannot fail
  d_common->attach(this);
}

template <typename T>
template <typename U>
assign_ptr<T>::assign_ptr                  // a kind of copy constructor
(assign_ptr<U> const & pointer,            // 'Y' is the "cast to" type
 xeona::polymorphic_cast_tag)              // tag 'struct' defined earlier
  throw(std::bad_cast):
   d_resource(dynamic_pointer_cast<T>(pointer.d_resource)),
   d_common(pointer.d_common)
{
  Deport deport("assign_ptr", "polymorphic cast ctor");
  if ( d_resource == 0 )                   // the cast failed
    {
      if ( xeona::nopro == false )         // meaning option '--krazy' not applied
        {
          std::ostringstream oss;
          oss << "'xeona::assign_ptr<T>' 'dynamic_pointer_cast' failed"
              << ", resource = " << pointer.d_resource.get();
          throw xeona::bad_assign_cast(oss.str());
        }
      d_common = shared_ptr<Common>(new Common());
    }
  d_common->attach(this);
}

// REPORT FUNCTION

// report details
template <typename T>
std::string
assign_ptr<T>::report
(const std::string& identifier,            // note default
 const std::string& comment,               // note default
 const unsigned     leftIndent) const      // note default
{
  std::string rsctype       = "(empty shared_ptror null resource)";
  if ( d_resource ) rsctype = xeona::demangle(typeid(*d_resource).name());

  const std::string fill(leftIndent, ' ');
  std::ostringstream oss;
  oss << fill << "assign_ptr details :";
  if ( !comment.empty() ) oss << " " << comment  << "\n";
  else                    oss << "\n";
  oss
    << fill << "  identifier : " << identifier                                  << "\n"
    << fill << "  resource   : " << d_resource.get()                            << "\n"
    << fill << "  ptr type   : " << xeona::demangle(typeid(T).name())           << "\n"
    << fill << "  rsc type   : " << rsctype                                     << "\n"
    << fill << "  revamps    : " << revamp_count()                              << "\n"
    << fill << "  clients    : " << client_count()                              << "\n"
    << fill << "  externals  : " << external_count()                            << "\n";
  return oss.str();
}
```

```
// CODE DEVELOPMENT CALLS

// get
// no null protection, unlike 'operator->'
template <typename T>
T*
assign_ptr<T>::get() const
{
  return d_resource.get();
}

// get assign_ptr
template <typename T>
assign_ptr<T>*
assign_ptr<T>::get_assign()
{
  return this;
}

// DOWNCAST CONSTRUCTORS

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::const_assign_cast <>
// ----------------------------------------------------------
//  Description  : wrapper to modified copy constructor
//  Role         : support for downcast functionality
//  Techniques   : tag struct, 'const_cast'
//  Status       : complete
// ----------------------------------------------------------

template <typename T, typename U>
assign_ptr<T>
const_assign_cast
(assign_ptr<U> const & pointer)
{
  Deport deport("free func", __func__);
  return assign_ptr<T>(pointer, xeona::const_cast_tag());
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::dynamic_assign_cast <>
// ----------------------------------------------------------
//  Description  : wrapper to modified copy constructor
//  Role         : support for downcast functionality
//  Techniques   : tag struct, 'dynamic_cast', return zero on cast fail
//  Status       : complete
// ----------------------------------------------------------

template <typename T, typename U>
assign_ptr<T>
dynamic_assign_cast
(assign_ptr<U> const & pointer)
{
  Deport deport("free func", __func__);
  return assign_ptr<T>(pointer, xeona::dynamic_cast_tag());
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::polymorphic_assign_cast <>
// ----------------------------------------------------------
//  Description  : wrapper to modified copy constructor
//  Role         : support for downcast functionality
//  Techniques   : tag struct, 'dynamic_cast', throw on cast fail
//  Status       : complete
//
//  Design notes
//
//      Unlike dynamic_assign_cast, the function throws a
//      'std::bad_cast' on failure.
//
//      See Karlsson (2006 pp54-61) for a discussion of
//      polymorphic casting.
//
// ----------------------------------------------------------

template <typename T, typename U>
assign_ptr<T>
polymorphic_assign_cast
(assign_ptr<U> const & pointer)
  throw(std::bad_cast)
```

```
  {
    Deport deport("free func", __func__);
    return assign_ptr<T>(pointer, xeona::polymorphic_cast_tag());
  }

} // namespace 'xeona'

#endif // _XEONA_PTR_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : glpkviz.h
//  file-create-date : Thu 05-Jun-2008 14:02 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : HTML visualization of GLPK problem instances / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/d/glpkviz.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This 'GlpkViz' class is coded for two roles: for use within
//  'xeona' and as a stand-alone unit.
//
//    the 'xeona' variant:
//
//      * takes a 'SolverIf' object (raw or shared_ptr)
//      * requires 'XE_GLPKVIZ_ALONE' = 0
//      * unit tests with 'glpkviz.ut0.cc'
//      * builds with the bash script 'mach'
//
//    the stand-alone variant:
//
//      * takes a GLPK 'glp_prob' instance
//      * requires 'XE_GLPKVIZ_ALONE' = 1
//      * unit tests with 'glpkviz.alone.ut.cc'
//      * builds with the bash script 'glpkviz.alone.sh'
//      * does not require units 'common' or 'logger' EXCEPT
//        for unit testing
//
//  Both require the same generic 'xeona' 'makefile' to be
//  present.
//
//  BROWSER PERFORMANCE VERSUS PROBLEM SIZE
//
//  GLPK viz cannot display large problems.  It was written to
//  assist the debugging of small-scale test problems.
//
//  For instance, using a modest system as follows:
//
//    system   : Linux 2.6.17 / Ubuntu GNU/Linux 6.10 (edgy) x86 / released Oct-2006
//    hardware : 1.4GHz Intel Celeron M / 512MiB RAM / 40GB HDD / 1024x768 / new Aug-2004
//    browser  : firefox 2.0.0
//
//  A 500x400 problem takes firefox about 40 seconds to display
//  and consumes an extra 110MB of memory, that is, over and
//  above an empty browser.

//  HEADER GUARD

#ifndef _GLPKVIZ_H_
#define _GLPKVIZ_H_
```

```
//  INTERNAL MACRO SETTING

#if !defined(XE_GLPKVIZ_ALONE)              // gives possibilty of external override
# define XE_GLPKVIZ_ALONE 0                 // 1 = stand-alone code, 0 = xeona
#endif

// check the macro setting (and allow for additional statements)

#if   (XE_GLPKVIZ_ALONE == 0)
#elif (XE_GLPKVIZ_ALONE == 1)
# warning "XE_GLPKVIZ_ALONE macro set to stand-alone (this may be what you want)"
#else
# error "XE_GLPKVIZ_ALONE macro not set to a supported value {0,1}"
#endif

//  LOCAL AND SYSTEM INCLUDES

#if   (XE_GLPKVIZ_ALONE == 0)
#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)
#endif // XE_GLPKVIZ_ALONE

#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <glpk.h>             // GNU GLPK mixed integer linear (MILP) solver

//  GLPK VERSION CHECK

// the following check assumes that a change to major version 5
// may also break this unit
#if ( GLP_MAJOR_VERSION == 4 && GLP_MINOR_VERSION < 38 )
#warning "GlpkViz requires GLPK 4.37 or better (check <glpk.h>)"
#endif

namespace svif
{
  class SolverIf;                           // grants friendship
}

//  CODE

// ----------------------------------------------------------
//  CLASS           : GlpkViz
// ----------------------------------------------------------
//  Description  : visualize a GLPK problem instance using HTML
//  Role         : as required by the client code
//  Techniques   : GLPK APIs, hand crafted mark-up creation
//  Status       : working
//
//  Design notes
//
//      The following (quite modest) standards apply:
//
//          mark-up          : HTML 4.0 Transitional
//          styles           : CSS 1
//          non-ASCII chars  : HTML named entities, XML decimal entities
//          colors           : RRGGBB
//
//      Closing tags are not required for <tr> and <td>.
//
//      The generated mark-up has been run thru the HTML 'tidy'
//      utility (01-Sep-2005 version).  The utility warns about
//      (nothing serious, this is the 'infinity' glyph, which
//      renders fine in 'firefox' 2.0):
//
//          - unknown entity "&#8734"
//
//  Quick introduction -- stand-alone variant
//
//      int main()
//      {
//        // GlpkViz stack object
//        GlpkViz firefox("firefox");         // browser call required
//
//        // create empty GLPK problem instance
//        glp_prob* prob = glp_create_prob();
//        firefox(prob, "empty problem");     // comment passed to web page
//      }
```

```
//
// -------------------------------------------------------

class GlpkViz
{
  // LOCAL ENUMERATIONS

private:

  enum Embellishment          // to help set the appropriate CSS class attribute
    {
      none    = 0,            // means do nothing

      // <td> classes
      narrow,                 // omit left and right padding
      label_c,                // for name strings from GLPK, centred
      label_l,                // for name strings from GLPK, left aligned
      kind_iv,                // integer-valued
      kind_bv,                // binary-valued
      zero,                   // zero structural coefficient
      meta,                   // row and col information
      meta_2,                 // ditto but spanning
      result,                 // solver values
      goal,                   // solver objective function value
      optimal,                // optimality reached
      extra,                  // extra reporting including reduced costs
      left,                   // left align
      right,                  // right align
      span_2,                 // span two cells, also adds a 'colspan' attribute
      error,                  // potential error or omission

      // <p> classes
      helptext,               // help text
      endtext                 // final text
    };

#if   (XE_GLPKVIZ_ALONE == 1)               // stand-alone role

  // DISABLED

private:

  GlpkViz();                                // zero-argument constructor
  GlpkViz(const GlpkViz& orig);             // copy constructor
  GlpkViz& operator= (const GlpkViz& orig);  // copy assignment operator

  // CREATORS

public:

  GlpkViz
  (const std::string browserInvoke);          // can pass in browser options as well

  ~GlpkViz();

  // FUNCTION CALL OPERATOR

  void
  operator()
  (glp_prob*         prob,                  // GLPK problem instance
   const std::string comment         = "",
   const int         outputPrecision = 6,
   const bool        glpkRounding    = true);

#elif (XE_GLPKVIZ_ALONE == 0)               // 'xeona' role

  // DISABLED

private:

  // CAUTION: zero-argument constructor cannot be disabled
  GlpkViz(const GlpkViz& orig);             // copy constructor
  GlpkViz& operator= (const GlpkViz& orig);  // copy assignment operator

  // CREATORS

public:

  GlpkViz
  (const std::string browserInvoke = xeona::webbrowser);    // thus a zero-argument ctor
```

```
  ~GlpkViz();

  // FUNCTION CALL OPERATORS

  void
  operator()
  (const svif::SolverIf* si,                          // solver interface object
   const std::string     comment         = "",
   const int             outputPrecision = 6,
   const bool            glpkRounding    = true);

  void
  operator()
  (const shared_ptr<svif::SolverIf> si,               // solver interface object
   const std::string                comment         = "",
   const int                        outputPrecision = 6,
   const bool                       glpkRounding    = true);

#endif // XE_GLPKVIZ_ALONE

  // UTILITY FUNCTIONS

private:

  // main calls taken approximately in the order shown

  void resetData();
  void prepareFilename();
  void prepareTitle();
  void createHtml();
  void createHelp();                          // extended form of the normal file
  void saveHtml(const std::string filename);
  void displayHtml(const std::string filename);

  // HTML helpers (the char* variants are to protect against char* NULL returns)

  void raw(const std::string line);                               // inserts raw text
  void rem(const std::string comment);                            // <!-- .. -->
  void title(const std::string info);                             // <title> .. </title>
  void title(const char* info);
  void h1(const std::string header);                              // <h1> .. </h1>
  void h1(const char* header);
  void rule();                                                    // <hr>
  void p(const Embellishment e = none);                           // <p></p>
  void p(const char* copy, const Embellishment e = none);
  void p(const std::string copy, const Embellishment e = none);
  void line();                                                    // <tr>
  void cell(const Embellishment e = none);                        // <td>
  void cell(const char* label, const Embellishment e = none);
  void li(const std::string item);                                // <li> .. </li>
  template <typename T> void cell (const T data, const Embellishment e = none);

  // "absent" GLPK glp functions

  double getConCoef(int row, int col);       // because no 'glp_get_mat_val'

  // utility functions

  std::string timestamp();                   // current time, duly formatted

  // INTERNAL DATA

private:

  std::string           d_browser;           // browser invocation string
  glp_prob*             d_prob;              // pointer to GLPK C struct
  std::ostringstream    d_html;              // HTML buffer before writing to file
  std::string           d_comment;           // optional from function call operator
  std::string           d_filename1;         // filename for normal version
  std::string           d_filename2;         // filename for extended help version
  std::string           d_title;             // for <h1> and also page <title>
  bool                  d_round;             // use LPX_K_ROUND near-to-zero rounding
  std::string           d_timestamp;         // single consistent timestamp for each html

  static std::string    s_helpExt;           // help file sub-extension
  static std::string    s_htmlExt;           // HTML file extension
  static int            s_callCount;         // track functor calls

#if   (XE_GLPKVIZ_ALONE == 0)                // 'xeona' role

  static logga::spLogger s_logger;           // shared_ptr to single logger object
```

```
#endif // XE_GLPKVIZ_ALONE

};

#endif // _GLPKVIZ_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : siglp.h
//  file-create-date : Tue 22-Apr-2008 14:37 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : semi-intelligent interface to GLPK MILP solver / header
//  file-status      : working (needs refactoring)
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/d/siglp.h $
//  HEADER GUARD

#ifndef _SIGLP_H_
#define _SIGLP_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../d/glpkviz.h"     // GLPK problem instance visualizer using HTML

#include <iostream>           // standard io
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <glpk.h>             // GNU GLPK mixed integer linear (MILP) solver

//  GLPK VERSION CHECK

// * this check assumes that major version 5 will also break the unit
// * the macros used are defined in 'glpk.h'
#if ( GLP_MAJOR_VERSION == 4 && GLP_MINOR_VERSION < 42 )
#warning "GLPK solver interface requires GLPK 4.42 or better (check <glpk.h>)"
#endif

//  MUST-KNOW INFORMATION
//
//      * semi-intelligent interface to the GNU GLPK optimization library
//      * tested with GLPK version 4.44 (06-May-2010)
//      * should be backward compatible to GLPK version 4.42 (13-Jan-2010)
//      * development began with GLPK version 4.25 (19-Dec-2007)
//      * read file 'DOCS/CODE.txt' for more information on GLPK interoperation
//      * employs one-based row and col indexing
//      * uses only the GLPK APIs documented in the relevant manual
//      * set in C++ namespace 'svif'
//      * search on "user-modifiable" for useful hardcoded settings
//      * structural variables are probably bounded to be non-negative
//      * 'stdout' for class reporting and GLPK output, 'stderr' for logging
//      * output suspend and resume calls could be useful when debugging
//
//  OVERARCHING NOTES
//
//  Purpose
//
//      The class 'SolverIf' offers a semi-smart interface to the
//      GNU Linear Programming Kit (GLPK) linear and
```

```
//      mixed-integer programming library.
//
//      The problem building and solver call methods provide a
//      higher-level interface than do the equivalent GLPK APIs.
//
//      A GLPK problem instance can be built on-the-fly.  In
//      other words, it is not necessary to specify the problem
//      size (row and column counts and coefficients estimate) or
//      problem klass (LP or MILP) in advance.
//
//      A number of data integrity checks are performed during
//      building and prior to the solver calls.
//
//      The appropriate solver calls are duly made and
//      interpreted.
//
//      While this code was written as part of the 'xeona'
//      application, it has been designed to be used as a
//      stand-alone unit and with or without the logging support
//      from other 'xeona' classes.
//
//  Extensiblity
//
//      The interface is meant to be easily extended to include
//      new problem building calls.
//
//  GLPK solver library required
//
//      This unit was tested using a statically linked GLPK
//      library archive.  The static archive was built using the
//      '-disable-shared' configure option.
//
//          $ cd glpk-0.00
//          $./configure --disable-shared
//
//  New-style GLPK APIs adopted
//
//      This unit uses the GLPK APIs as documented in the current
//      reference manual.  These are a mix of old- and new-style
//      and can be recognizable thus:
//
//          'lpx' functions migrating to 'glp'
//          'LPX' constants (often macros) migrating to 'GLP'
//
//      The new APIs are being progressively introduced from GLPK
//      release 4.17 (26-May-2007) (to the best of my knowledge)
//      and the process is still not complete as of 4.28
//      (25-Mar-2008).
//
//  Boost libraries are not used in this unit
//
//      For reasons of portability, this implementation does NOT
//      use Boost (nor TR1) libraries.  In particular, this means
//      no 'boost::shared_ptr<>' smart pointers.
//
//      The 'siglp' unit test for this unit does, however, use
//      Boost smart pointers, but it should be a straightforward
//      matter to revert these to raw pointers.
//
//      More importantly, the logger unit defined in "logger.h"
//      and 'logger_fwd.h' employs several Boost libraries.  If
//      the required libraries are not available then a new
//      logger unit will need to be created with either
//      substituted functionality or just plain disabled.
//
//      Consult the world-wide-web for more information on Boost
//      libraries.
//
//  Removal of support for 'logger' and 'common'
//
//      If need be, the headers "logger.h", "logger_fwd.h", and
//      "common.h" can omitted and the resulting compiler
//      warnings used to cleanse the code.  Alternatively, search
//      and delete lines matching 'logga' and then check the
//      surrounding construction.
//
//      A fake logger suite was written early on in this project.
//      The files 'logger_fake.h' and 'logger_fake.cc' would make
//      a good starting point for the development of a disabled
//      logger.
//
```

```
//  Problem labeling
//
//      The following labeling prefixes are currently hardcoded
//      at the start of 'siglp.cc':
//
//          pro-    problem
//          obj-    objective function
//          con0-   constraint equation
//          var0-   strictly real-valued (continuous) structural variable
//          vaz0-   strictly integer-valued (integer) structural variable
//          vab0-   strictly 0-1-valued (binary) structural variable
//
//  Terminal output, redirection, and suppression
//
//      Note that standard output or 'stdout' is represented by
//      UNIX file descriptor 'STDOUT_FILENO' (set to 1) and
//      standard error or 'stderr' is represented by UNIX file
//      descriptor 'STDERR_FILENO' (set to 2).  And that, for C++
//      ouput streams, 'std::cout' maps to 'stdout' and
//      'std::cerr' and 'std::clog' both map to 'stderr'.
//      Moreover, 'std::clog' is the buffered form of 'std::cerr'
//      and 'std::cout is also buffered.  In addition, the
//      insertion of 'std::endl' and 'std::flush' both flush the
//      streams to which they apply.
//
//      GLPK output > 's_os' > cout > stdout
//
//      GLPK writes to 'stdout' by default, but this can be
//      changed by defining a custom hook function and then
//      calling 'glp_term_hook'.  This is done here, with hook
//      function named 'termHook'.  The new stream (a pointer in
//      fact) is named 's_os' and initially maps to 'std::cout'.
//
//      Class native reporting > 's_os' > cout > stdout
//
//      Class native reporting is sent to the same 's_os' as
//      GLPK output.
//
//      Logger output > 'd_os' > clog > stderr
//
//      The logger currently outputs to 'std::clog'.  This
//      behavior can be overridden by explicitly defining the
//      internal-use preprocessor macro 'XE_OUTPUT', set in
//      'logger.cc'.
//
//      'xeona' output > cout > stdout
//
//      For completeness, the initial 'xeona' splash screen
//      and final information screen are written to
//      'std::cout'.
//
//      Redirection
//
//      The 's_os' stream can be simply dumped using the
//      'outputSuspend' and 'outputResume' calls.  Or
//      redirected to a log file by modifying the relevant
//      code in the free function 'svif::refDumpStream' in
//      'siglp.cc'.
//
//      The call 'stdoutRedirect' can also be used, but it
//      effect is application-wide and irreversible (as
//      currently implemented).  This call simply and crudely
//      closes UNIX file descriptor 'STDOUT_FILENO'.
//
//      Motivation
//
//      The reason that effort has been put into controlling
//      both the level of reporting and the output direction
//      is that the amount of text produced can easily become
//      astronomical.
//
//  File organization
//
//      This file is organized roughly as follows:
//
//        * public enumerations
//        * constructor and destructor
//        * GLPK clean up (free GLPK environment)
//        * output controls
//        * solver preference setting
//        * problem building
```

```
//        * solver invocation
//        * solution recovery
//        * general information
//        * miscellaneous
//        * utility (private) functions
//        * instance internal data
//        * class-wide (static) internal data
//
//  Preferred terminology (because it varies so much)
//
//      objective function (not goal function)
//      constraint matrix (not structural matrix)
//      mixed-integer linear program (not mixed-integer program)
//      right-hand side bound
//      problem klass (intentionally 'k' to distinguish it from the C++ keyword)
//      GLPK problem instance (not GLPK problem object)
//
//  GLPK integer solvers
//
//          glp_intopt() : prior optimal LP relaxation required
//

//  ADDITIONAL NOTES (r2718) : GLPK now has new reporting controls as follows:
//
//  Solver level control
//
//      Three forms
//
//          smcp  = simplex solver control parameters
//          iptcp = interior-point solver control parameters
//          iocp  = integer optimization [branch and cut] control parameters
//
//      Data and functions
//
//          structs:
//
//              glp_smcp
//              glp_iptcp
//              glp_ipcp
//
//          reporting level data member:
//
//              all contain data member 'msg_lev', defaulting to 'GLP_MSG_ALL'
//
//          struct intitialization calls:
//
//              int  glp_init_smcp (&parm)
//              int  glp_init_iptcp(&parm)
//              void glp_init_iocp (&parm)
//
//          solver calls:
//
//              int glp_simplex (glp_prob*  lp, const glp_smcp*  parm)
//              int glp_exact   (glp_prob*  lp, const glp_smcp*  parm)
//              int glp_interior(glp_prob*   P, const glp_iptcp* parm)
//              int glp_intopt  (glp_prob* mip, const glp_iocp*  parm)
//
//          macro definitions for use with 'msg_lev:'
//
//              GLP_MSG_OFF : no output
//              GLP_MSG_ERR : error and warning messages only
//              GLP_MSG_ON  : normal output
//              GLP_MSG_ALL : full output (including informational messages)
//
//      Representative code:
//
//          glp_smcp parm;                // simplex solver control parameters
//          glp_init_smcp(&parm);         // initialize with defaults
//          parm.meth    = GLP_DUAL;
//          parm.msg_lev = GLP_MSG_ERR;   // modify setting
//          glp_simplex(P, &parm);
//
//  Environment-level control
//
//      API routines:
//
//          glp_term_out  : enable/disable terminal output
//          glp_term_hook : intercept terminal output
//
//  Note also (does not control reporting)
//
```

```
//      bfcp = basic factorization control parameters

//  CODE

namespace svif                              // CAUTION: note namespace
{
  // ------------------------------------------------------------
  //  CONSTANT        : DBUG
  // ------------------------------------------------------------

  extern const bool DBUG;                   // set via _XDEBUG macro in 'siglp.cc'

  // ------------------------------------------------------------
  //  ENUM            : ProblemStatus
  //  ENUM            : ObjectiveSense
  //  ENUM            : SolverType
  //  ENUM            : ProblemKlass
  //  ENUM            : ColumnKind
  //  ENUM            : ConstraintSense
  //  ENUM            : SolutionStatus
  // ------------------------------------------------------------
  //  Description  : supporting enums, often close to GLPK macros
  //  Role         : to improve code safety and readability
  //  Status       : complete
  //
  //  Design notes
  //
  //     These enumerations are mostly self-explanatory.  The
  //     'other' values are provided for testing purposes.  Some
  //     further clarification is given below:
  //
  //       ColumnKind
  //
  //           the 'ColumnKind' enum is no longer required after
  //           GLPK introduced 'GLP_BV', but is nonetheless
  //           retained for completeness
  //
  //       ObjectiveSense
  //
  //           although GLPK defaults to maximize, this class
  //           requires the objective sense to be set explicitly
  //
  //       soln_not_usable
  //
  //           means the current solution is infeasible, but has
  //           not been proven to be generally so (this latter
  //           condition is sometimes known as 'nonfeasible')
  //
  //       soln_proven_bad
  //
  //           means the current problem has been proven to have
  //           no feasible solution (that is, it is 'nonfeasible')
  //
  // ------------------------------------------------------------

  enum ProblemStatus                // meaning the GLPK problem instance status
    {
      status_not_specified = 0,
      status_incomplete    = 1,     // condition below does not hold
      status_empty         = 2,     // objective sense set but still empty
      status_solvable      = 3,     // objective sense and at least one row, col, and coeff
      status_submitted     = 4,     // submitted to solver
      status_other         = 9
    };

  enum ObjectiveSense               // termed "objective direction" in GLPK
    {
      sense_not_specified  = 0,
      minimize             = 1,     // GLP_MIN
      maximize             = 2      // GLP_MAX (maximize is the GLPK default)
    };

  enum SolverType
    {
      solver_not_specified = 0,
      solver_simplex       = 1,     // 'glp_simplex' simplex method
      solver_network       = 2,     //  CAUTION: network simplex planned for late-2008
      solver_interior      = 3,     // 'glp_interior' interior point method
      solver_integer       = 4,     // 'glp_intopt' integer method
      solver_other         = 9
    };
```

```
enum ProblemKlass                // LPX_LP, LPX_MIP, 'lpx_get_class' no longer supported
  {
    prob_not_specified   = 0,
    prob_linear          = 1,    // GLP_CV
    prob_mixed_integer   = 2,    // GLP_CV + GLP_IV (+ GLP_BV optionally)
    prob_mixed_zero_one  = 3,    // GLP_CV           + GLP_BV
    prob_pure_integer    = 4,    //          GLP_IV (+ GLP_BV optionally)
    prob_pure_zero_one   = 5,    //                   GLP_BV
    prob_other           = 9
  };

enum ColumnKind                  // these classifications are mutually exclusive
  {
    col_not_specified    = 0,
    col_continuous       = 1,    // GLP_CV, strictly continuous
    col_integer          = 2,    // GLP_IV, strictly integer
    col_binary           = 3,    // GLP_BV, strictly binary
    col_other            = 9
  };

enum ConstraintSense             // same terminology as the IBM OSI project
  {
    con_not_specified    = 0,
    L                    = 1,    // <= constraint
    E                    = 2,    //  = constraint
    G                    = 3,    // >= constraint
    R                    = 4,    // ranged constraint
    N                    = 5     // free constraint
  };

enum SolutionStatus
  {
    soln_undefined       = 0,
    soln_optimal         = 1,    // solution optimal
    soln_feasible        = 2,    // solution usable but not optimal
    soln_not_usable      = 3,    // solution not usable but problem not proven bad
    soln_proven_bad      = 4,    // problem proven bad
    soln_faulty          = 5,
    soln_other           = 9
  };

// ------------------------------------------------------------
//  ENUM            : ReportingLevel
// ------------------------------------------------------------
//  Description  : supporting enum to control reporting
//  Status       : work in progress
//
//  Design notes
//
//      First be aware that logging verbosity is controlled via
//      the 'Logger' class and is thus not managed by this
//      class.  That said, much of the code here contains
//      'logga::Logger::repx' calls.
//
//      GLPK native output control works on three levels:
//
//          * at the GLPK solver call level via the control
//          parameter 'msg_lev' which defaults to 'GLP_MSG_ALL'
//          -- as of 4.33, supported calls comprise
//          'glp_simplex', 'glp_intopt', and 'glp_interior'
//
//          * at the GLPK problem instance level via
//          'lpx_set_int_parm' as follows:
//
//              lpx_set_int_parm(d_prob, LPX_K_MSGLEV, value)
//
//          * at the GLPK library environment level, using the
//          calls 'glp_term_out' and 'glp_term_hook' -- for
//          example, 'glp_term_out(GLP_OFF)' completely blocks
//          all normal terminal output (but NOT any intercepted
//          and then redirected output) and applies globally
//
//      These last two levels are also mirrored in the
//      'SolverIf' class as follows:
//
//          * the value of object variable 'd_noise' of type
//          'ReportingLevel' determines how the GLPK instance
//          is initiated -- 'd_noise' is optionally set in the
//          constructor, otherwise it defaults to svif::low and
//          svif::high depending on whether svif::DBUG is false
```

```
//          or true
//
//          * output suspend is controlled via the static
//          functions 'outputSuspend' and 'outputResume' and
//          the current status is recorded by 's_noise'
//
//      Some class native reporting -- mostly in the form of
//      information blocks -- also exists and controlled via
//      'd_noise' and sometimes dependent as well on the
//      solution status.
//
//      The GLPK settings (old and new style) are as follows:
//
//          LPX_K_MSGLEV = 0 or GLP_MSG_OFF - no output (although leakage may occur)
//          LPX_K_MSGLEV = 1 or GLP_MSG_ERR - errors and warnings only
//          LPX_K_MSGLEV = 2 or GLP_MSG_ON  - normal
//          LPX_K_MSGLEV = 3 or GLP_MSG_ALL - normal plus information
//
//      The static function 'stdoutRedirect' simply redirects
//      or disables 'stdout', depending on whether the call was
//      passed a logfile name or not.  As currently
//      implemented, this call is irreversible.  Deploy it with
//      care (indeed, this function should probably not be
//      offered).
//
//  Summary
//
//      The 'ReportingLevel' emum applies only to 'SolverIf'
//      instances and controls both GLPK output and class
//      native reporting.
//
//      The static calls 'outputSuspend/Resume()' disable and
//      enable GLPK output and class native reporting on a
//      class-wide basis.
//
//      Logging verbosity is controlled via the 'Logger' class
//      and is not managed here.
//
//      'Stdout' across the entire application can be
//      permanently redirected or closed by calling
//      'stdoutRedirect'.
//
// ------------------------------------------------------------

enum ReportingLevel
  {
    not_specified  = 0,
    silent         = 1,     // GLPK no output (although leakage may occur)
    low            = 2,     // GLPK errors and warning
    medium         = 3,     // GLPK normal             + problem info
    high           = 4      // GLPK normal plus info   + problem info plus GLPK report
  };

// ------------------------------------------------------------
//  FREE FUNCTION   : refDumpStream
// ------------------------------------------------------------

std::ofstream&
refDumpStream();                          // used by 'outputSuspend' and 'outputResume'

// ------------------------------------------------------------
//  CLASS           : SolverIf
// ------------------------------------------------------------
//  Description  : semi-intelligent interface to GLPK solver
//  Role         : offers higher-level calls when building and solving MILP problems
//  Status       : complete (more or less)
//
//  Design notes
//
//      Row and col indexing is one-based, that is, the
//      indexing starts from one (however, the objective
//      function 'shift' is assigned index zero).
//
//      Significant additional notes are contained in the code.
//
//  CAUTION: friendship and included headers
//
//      Be aware that friendship introduces some tighter
//      requirements on the order in which the compiler see
//      things.
//
```

```
//  CAUTION: freeing the GLPK environment
//
//      Normally a call to 'freeGlpkEnvironment' is not needed
//      because the final 'SolverIf' destructor call will do
//      this.
//
//      If required, the static function 'freeGlpkEnvironment'
//      should only be invoked near the end of the 'main'
//      function because the call will dump any and all
//      existing GLPK problem instances.
//
//  References (GLPK 4.25)
//
//      Makhorin, Andrew.  2007.  GNU linear programming kit :
//        reference manual version 4.25 -- Draft edition
//        (December 2007).  Free Software Foundation, Boston, MA,
//        USA.  [author affiliation: Department for Applied
//        Informatics, Moscow Aviation Institute, Moscow, Russia]
//        (distributed with the GLPK source code as LaTeX, DVI,
//        and PostScript)
//
// ------------------------------------------------------

class SolverIf
{
  // ------------------------------------------------------
  // FRIENDS

  friend void GlpkViz::operator() (const SolverIf*, std::string, int, bool);

  // ------------------------------------------------------
  // DISABLED

private:

  SolverIf();                                   // zero-argument constructor
  SolverIf(const SolverIf& orig);               // copy constructor
  SolverIf& operator= (const SolverIf& orig);   // copy assignment operator

  // ------------------------------------------------------
  // CREATORS

public:

  SolverIf
  (std::string    tag,                                // problem name tag, mandatory
   ReportingLevel noise = svif::not_specified);       // default chosen by _XDEBUG

  ~SolverIf();                              // no need to be virtual

  // ------------------------------------------------------
  // GLPK CLEAN-UP : a class-wide call, CAUTION: no longer
  // needed in client code because it is called when the
  // 'SolverIf' destructor is invoked on the last instance --
  // but if employed, use only near the end of the 'main'
  // function because any existing GLPK problem instances will
  // be unceremoniously "invalidated"

  static
  bool                                      // 'false' if problems encountered
  freeGlpkEnvironment();                    // calls 'glp_free_env', warns if instances

  // ------------------------------------------------------
  // REPORTING CONTROL : class-wide calls controlling GLPK
  // output and native reporting -- as opposed to xeona-style
  // logging
  //
  // these calls may be made before any 'SolverIf' objects have
  // be created

  // HEALTH WARNING: 'stdoutRedirect' is application-wide and permanent!

  static
  void
  stdoutRedirect                            // CAUTION: see above health warning
  (const std::string logname = "");         // default will simply close 'stdout'

  static
  void
  outputSuspend();                          // disable all GLPK and class native output
```

```
    static
    void
    outputResume();                         // re-enable all GLPK and class native output

    // ------------------------------------------------------
    // SOLVER PREFERENCES
    //
    // CAUTION: first note that the concepts of "simplex
    // presolve" and "MILP initial LP solve" are quite different
    // and distinct -- the function 'initUseSimpPresolver'
    // declared below covers the former

    void
    initSetPrefLPSolver                     // advisory only
    (SolverType solverType);                // preferred LP solver

    void
    initEmployScaling();                    // the GLPK default is not to scale

    void
    initEmployAdvBasis();                   // the GLPK default is not to construct this

    void
    initSimplexPresolver();                 // the GLPK default is not to presolve

    void
    initIntegerPresolver();                 // the GLPK default is not to presolve

    // ------------------------------------------------------
    // PROBLEM BUILDING : underlying model, calls, and rules
    //
    //  Users should read this comment block carefully.
    //
    //  This implementation uses dynamic containers, tracks the
    //  problem klass (LP or MILP) "on-the-fly", and
    //  automatically determines which calls and checks are to
    //  make at solve-time.  No up-front problem size and klass
    //  specifications and estimates are required.
    //
    //  In relation to the constraint matrix and associated
    //  objective and rhs vectors:
    //
    //      i is the one-based row count
    //      j is the one-based col count
    //
    //  But note also that the 'shift' objective coefficient
    //  resides at index 0.
    //
    //  These calls assume the problem is given in 'ordinary'
    //  rather than 'standard' form (see GLPK documentation
    //  chapter 1 "Introduction" and appendix C "CPLEX LP
    //  format").  Thus for structural variable x_j and
    //  constraint equation i (or simply row i and col j):
    //
    //      obj_j                           // objective coefficient j
    //      coeff_ij                        // constraint coefficient i,j
    //      <=|=|>=                         // constraint sense i
    //      rhs_i                           // rhs bound i
    //      x_j optionally in { int, bin }  // additional restriction
    //
    //  plus (as a 'SolverIf' default) a global condition that
    //  structural variables be non-negative (this requirement
    //  derives from 'xeona' and not GLPK)
    //
    //      x_j   >= 0                      // global non-negativity condition
    //
    //  and also note that, unless otherwise specified:
    //
    //      'value' is of fundamental type double and can be zero
    //          or strictly negative
    //
    //  The default col bounds are controlled by the internal
    //  macro 'XE_BNDS_DEFAULT'.  This macro is set in the
    //  implementation file.  If a single col requires finer
    //  bounds then two approaches exist.  Add a unitary entry
    //  constraint row for each requisite bound.  Or call
    //  'reviseBnds' directly.  Unlike the first method, a call
    //  to 'reviseBnds' allows the existing col bounds to be
    //  widened.  The choice of method depends somewhat on
    //  whether the "restriction" arises as a normal problem
    //  constraint or as something more fundamental.
```

```
//
//  The 'markVarBinary' call sets the col bounds to [0,1] and
//  the other mark calls return the bounds to the default set
//  by 'XE_BNDS_DEFAULT'.  Hence, col bounds are consistent
//  and universal (aside from the toggling effect for binary
//  variables).
//
//  A row, col, objective, or problem 'label' is formed from
//  the appropriate fixed-character prefix (defined in the
//  implementation file), any index data (or the timestamp,
//  in the case of a problem label), and the optionally
//  supplied 'tag' if given.
//
//  Label-setting is also used to enforce call order and to
//  detect overwrite conditions.
//
//  A 'complete' problem must have at least one row, one col,
//  and one constraint coefficient set, and have had an
//  explicit call to 'setObjectiveSense' (in other words,
//  unlike GLPK, there is no "maximize" default).  Otherwise
//  the problem will be deemed 'incomplete' and a call to
//  'runSolver' will see it abandon its task after some
//  preliminary checks.
//
//  The following problem building functions are provided:
//
//      setProblemLabel
//
//          * takes    : tag (mandatory for this call)
//          * returns  : void
//          * logs     : nothing
//          * relabels : problem (first set by the constructor)
//
//      setObjectiveSense
//
//          * takes    : objective sense {maximize, minimize}
//          * optional : tag
//          * returns  : void
//          * action   : sets objective sense
//          * logs     : overwrite calls
//          * required : for problem completeness (else solve-time failure)
//          * labels   : associated objective function
//
//      setObjectiveLabel
//
//          * takes    : tag (mandatory for this call)
//          * returns  : void
//          * logs     : nothing
//          * relabels : objective function
//
//      incShift
//
//          * takes    : const term and loads incrememental
//          * returns  : void
//          * logs     : nothing
//          * labels   : nothing
//
//      loadObj
//
//          * takes    : col (zero for "shift"), value (can be zero)
//          * optional : tag
//          * returns  : void
//          * action   : loads obj_j
//          * action   : applies structural variable 'XE_BNDS_DEFAULT' condition
//          * logs     : non-progressive calls, overwrite calls
//          * required : one call for problem completeness (else solve-time failure)
//          * note     : unset cols are deemed zero by GLPK
//          * labels   : associated structural variable
//
//      loadRhs
//
//          * takes    : row, value (can be negative), constraint sense {L, E, G}
//          * optional : tag
//          * returns  : void
//          * action   : loads rhs_i
//          * action   : sets the constraint sense i thus:
//                           svif::L is <=
//                           svif::E is  =
//                           svif::G is >=
//          * logs     : non-progressive calls, overwrite calls
//          * required : one call for problem completeness (else solve-time failure)
```

```
//          * note     : ranged constraints require two separate calls (like CPLEX)
//          * labels   : associated constraint equation
//
//      loadCof
//
//          * takes    : row, col, value (zero is legal but rejected)
//          * returns  : 'false' if zero value, otherwise 'true'
//          * action   : loads coeff_ij
//          * ignores  : zero values (which GLPK will also ignore)
//          * logs     : zero values, overwrite calls
//          * required : one call for problem completeness (else solve-time failure)
//          * labels   : nothing to label
//
//      reviseBnds
//
//          * takes    : col, lower value, upper value
//          * returns  : void
//          * action   : sets bounds for the nominated structural variable
//          * logs     : premature calls, overwrite calls
//          * required : strictly optional
//          * note     : will take 'inf' or '-inf' as an IEEE 754 (IEC 60559) infinity
//          * note     : 'XE_BNDS_DEFAULT' actioned under 'loadObj'
//          * labels   : nothing to label
//
//      markVarInteger
//
//          * takes    : col
//          * returns  : void
//          * action   : marks structural variable j as integer-valued
//          * action   : returns col bounds as per 'loadObj'
//          * logs     : premature calls, overwrite calls
//          * warns    : non-integer bounds (GLPK will later reject problem)
//          * relabels : associated structural variable
//
//      markVarBinary
//
//          * takes    : col
//          * returns  : void
//          * action   : marks col j as 0-1 binary
//          * logs     : premature calls, overwrite calls
//          * relabels : associated structural variable
//
//      markVarContinuous
//
//          * takes    : col
//          * returns  : void
//          * action   : marks col j as continuous (the default in any case)
//          * action   : returns col bounds as per 'loadObj'
//          * logs     : premature calls, overwrite calls
//          * note     : the default is continuous and this call is not normally used
//          * relabels : associated structural variable
//
//      reviseObj
//          * takes    : col (zero for "shift"), value (can be zero)
//          * omits    : tag (current value is retained)
//          * returns  : previous objective value
//          * action   : reloads obj_j
//          * logs     : may or may not log all calls (read the code)
//          * required : no
//          * note     : should follow 'loadObj' call
//          * note     : does not change tag, column kind, col bounds
//          * labels   : associated structural variable
//
//      resetProblem
//
//          * takes    : nothing
//          * returns  : void
//          * action   : removes built data and warm start info, retains problem label
//          * logs     : none
//
//      recreateProblem (provided for test purposes only)
//
//          * optional : tag
//          * returns  : void
//          * action   : GLPK problem object is deleted and recreated
//          * logs     : none
//          * relabels : if passed tag, relabels problem, else recycles existing label
//
//  The solver call also performs some problem build
//  integrity checks.
//
```

```
//  The following call and call order rules apply:
//
//      * the above calls -- except the three variable
//      marking calls and 'reviseBnds' -- can be made in any
//      order, need not be row and col incremental (termed
//      "progressive" here), and can repeat earlier calls
//      with either identical or different data (termed
//      "overwrite" here)
//
//      * notwithstanding, GLPK requires a complete problem
//      at solve-time
//
//      * a call to 'markVarInteger' or 'markVarBinary' (or
//      'markVarContinuous' for that matter) MUST be made
//      after the relevant 'loadObj' call and may NOT be made
//      in advance.
//
//      * a call to 'reviseBnds' or 'reviseObj' MUST be made
//      after the relevant 'loadObj' call and may NOT be made
//      in advance.
//
//      * a non-progressive call to 'loadRhs' or 'loadObj' is
//      legal, but will generate a log message regarding a
//      'skip' or 'backfill' occurrence as appropriate
//
//      * an overwrite call (with identical or different
//      data) is legal, but also will generate debug output
//
//      * if a structural variable is declared binary, any
//      prior or subsequent bounds settings will be ignored
//
//      * if a structural variable is declared integer, any
//      prior or subsequent bounds settings must be strictly
//      integer-valued (value == floor(value))
//
//      * on completion, all rows and columns must be
//      explicitly specified -- there cannot be "gaps"
//
//  The minimum specification is the constraint sense and (at
//  least) one row, one col, and one constraint coefficient.
//
//  As currently implemented, warm starting is not supported.
//  A call to 'resetProblem' will remove all row and col LP
//  status information.  An alternative method 'zeroProblem'
//  which retained such information was considered but
//  rejected because of the amount of new code needed.
//  Hence, this solver interface does not presently offer
//  warm start functionality.  Unless huge problems are being
//  modeled, is unlikely that warm starting would provide for
//  significant speed gains.

void
setProblemLabel                         // overwrite label set on construction
(const std::string     tag);            // problem name tag, no default provided

void
setObjectiveSense
(const ObjectiveSense  objSense,        // {minimize, maximize}
 const std::string     tag = "");       // objective function name tag, note default

void
setObjectiveLabel
(const std::string     tag);

void
incShift                                // increment the "shift"
(const double          constTermValue); // objective constant term value increment

void
loadObj
(const int             col,             // col index (can be zero)
 const double          objValue,        // objective coefficient value
 const std::string     tag = "");       // structural variable name tag, note default

void
loadRhs
(const int             row,             // row index
 const double          rhsValue,        // RHS constant value
 const ConstraintSense conSense,        // constraint sense {L, E, G}
 const std::string     tag = "");       // constraint equation name tag, note default
```

```
    bool                                // 'false' indicates 'coeffValue' was zero
    loadCof
    (const int             row,         // row index
     const int             col,         // col index
     const double          coeffValue); // structural coefficient value

    void
    reviseBnds
    (const int             col,         // col index
     const double          lowerValue,  // lower bound, can be '-inf'
     const double          upperValue); // upper bound, can be 'inf'

    void
    resetDefaultBnds
    (const int             col);        // col index

    void
    markVarInteger
    (const int             col);        // col index

    void
    markVarBinary                       // uses GLP_BV (introduced in GLPK 4.16)
    (const int             col);        // col index

    void
    markVarContinuous                   // the default, only required if revising
    (const int             col);        // col index

    double                              // prior value
    reviseObj
    (const int             col,         // col index (can be zero)
     const double          objValue);   // objective coefficient value

    void
    resetProblem();                     // empty the problem instance

    void
    recreateProblem                     // delete/recreate problem instance
    (std::string     tag = "");         // recycles prior label if tag is empty

    // -----------------------------------------------------
    // METHODS INTENDED FOR TESTING

    void
    convertToLP();                      // downgrade to LP if currently MILP

    void
    toggleObjective();                  // multiply current objective by minus one

    bool                                // return previous value
    setGlpkRounding                     // reporting uses GLPK close-to-zero rounding
    (const bool rounding);              // 'true' = reporting uses GLPK close-to-zero

    glp_prob*                           // dangerous after 'SolverIf' host destructed
    obtainGlpProbPtr();                 // to provide direct access

    // -----------------------------------------------------
    // PROBLEM INFORMATION : give the information as it stands,
    // calls can made before or during the problem build or
    // before or after the solver call

    std::string
    reportGlpkProblem();                // using GLPK calls as far as possible

    std::string
    getProblemLabel() const;

    ProblemKlass                        // an 'svif::' enumeration
    getProblemKlass() const;            // problem klass

    ObjectiveSense                      // an 'svif::' enumeration
    getObjectiveSense() const;          // objective sense, if set

    std::string
    getObjectiveSenseStr() const;       // interpreted

    std::string                         // with "obj-" (or similar) prefix
    getObjectiveLabel() const;

    std::string                         // without "obj-" (or similar) prefix
    getObjectiveTag() const;            // tag for use in future 'SolverIf' calls
```

```
    ColumnKind
    getVarKind
    (const int col) const;

    std::string                             // with "va[rzb]-" (or similar) prefix
    getVarLabel
    (const int col) const;

    std::string                             // without "va[rzb]-" (or similar) prefix
    getVarTag                               // tag for use in future 'SolverIf' calls
    (const int col) const;

    int
    getVarCount() const;                    // number of variables, includes blank cols

    int
    getConCount() const;                    // number of constraints, includes blank rows

    int
    getNonZeroCoeffCount() const;           // number of non-zero structural coefficients

    bool
    isUseSimpPresolver() const;             // note the default is set on construction

    bool
    isLP() const;                           // all variables are continuous

    double
    getLowerBnd                             // zero by default
    (const int col) const;

    double
    getUpperBnd                             // inf by default
    (const int col) const;

//      void
//      getFormattedProblem() const;            // not implemented, use 'GlpkViz' instead

    // ------------------------------------------------------
    // POST-BUILD MANIPULATION

    std::vector<double>                     // return current objective function
    getObjective() const;

    bool                                    // 'true' for success
    renewObjective
    (const std::vector<double> objFunc,     // entire function
     const std::string         tag = "");   // objective function name tag, note default

    bool                                    // 'true' for success
    zeroBarMeObjective
    (const int         nonzeroCol,          // nominate the only non-zero column
     const double      value,               // objective coefficient value (often 1.0)
     const std::string tag = "");           // new objective function name, note default

    // ------------------------------------------------------
    // SOLVER INVOCATION

    void
    runSolver();                            // sole point of entry

    // ------------------------------------------------------
    // SOLUTION RECOVERY : usage must follow solver call

    double
    getObjectiveValue() const;              // the objective function value

    double
    getVarValue                             // a structural variable value
    (const int col) const;

    int
    getIntegerVarValue                      // a structural variable value
    (const int col) const;

    bool                                    // 'false' = 0 and 'true' = 1 (like C++)
    getBinaryVarValue                       // a structural variable value
    (const int col) const;

    double
```

```
    getObjCoeff                             // an objective coefficient
    (const int col) const;

    double
    getObjCoeffXVarValue                    // getObjCoeff * getVarValue
    (const int col) const;

    double
    getSlackValue                           // reduced cost of an auxiliary variable
    (const int row) const;

    double
    getShadowValue                          // reduced cost of a structural variable
    (const int col) const;

    bool
    isUsableSoln() const;                   // is feasible plus other usefulness tests

    bool
    isOptimalSoln() const;                  // proven optimal

    std::string
    getFormattedResults                     // reports Z and x_j using formatted text
    (const std::string objectiveTag = "Z",
     const std::string variableTag  = "x");

    // ------------------------------------------------------
    // GENERAL INFORMATION

    const std::string                       // CAUTION: consider also 'getProblemLabel'
    getProblemTag() const;                  // as submitted on construction or revised

    int
    getAlertCount() const;                  // number of alerts

    void
    writeInfo                               // wrapper to take a C-string
    (const char* filestem) const;           // filestem name, extensions added later

    void
    writeInfo() const;                      // wrapper to take current label as filestem

    void
    writeInfo                               // working function, calls GLPK info routines
    (const std::string filestem) const;     // filestem name, extensions added later

    std::string                             // gives "yes" or "no"
    yesOrNo
    (const bool trueOrFalse) const;         // used to reinterpret bool-returning calls

    static
    const std::string                       // string format 0.00
    getGlpkVersion();                       // calls 'glp_version'

    // ------------------------------------------------------
    // MISCELLANEOUS

    double
    getInf() const;                         // returns IEEE 754 (IEC 60559) infinity

    double
    getNaN() const;                         // returns IEEE 754 (IEC 60559) not-a-number

    // ------------------------------------------------------
    // UTILITY FUNCTIONS

  private:

    // problem creation methods -- called in the sequence given

    void
    utilCreateGlpkProb
    (const std::string tag = "");

    void
    utilSetGlpkDefaults();

    const std::string                       // recover the problem label
    utilDeleteGlpkProb();

    // support methods
```

```
  void
  utilSetDefaultBnds                        // controlled by 'XE_BNDS_DEFAULT' macro
  (int col);

  void
  utilResetCoeffVectors();                  // reset the three sparse matrix vectors

  void
  utilResetObjectData();                    // reset the relevant status variables

  // solver methods

  void
  solverCheckProb();                        // only issues logging messages

  void
  solverInvokeSolver();

  void
  solverAnalyzeSoln();

  std::string
  utilReportOnProb();                       // passive and hence optional

  std::string
  utilReportOnSolver();                     // passive and hence optional

  std::string
  utilReportOnSoln();                       // passive and hence optional

  // housekeeping methods

  int                                       // current alert count
  incrementAlertCount();                    // alerts mostly arise from data issues

  // output formatting methods used by reporting utilities

  template <typename T>                     // implicit instantiation usually okay
  inline                                    // compiler hint
  std::string                               // output format depends on implementation
  numToString                               // number to string conversion
  (const T number) const;                   // usually int or double

  inline                                    // compiler hint
  std::string                               // output format depends on implementation
  indexToString                             // index to string conversion
  (const int index) const;

  void
  formatLineOutput                          // for standard reporting of text
  (std::ostringstream& ssOut,
   const std::string   tag,
   const std::string   value);              // used for strings

  template <typename T>                     // implicit instantiation usually okay
  void
  formatLineOutput                          // for standard reporting of numbers
  (std::ostringstream& ssOut,
   const std::string   tag,
   const T             value,               // used for ints and doubles
   const std::string   extra = "");         // optional trailing string

  void
  updateProbStatus();                       // update 'd_probStatus' data member

  void
  updateProbKlass();                        // update 'd_probKlass' data member

  // ========================================================
  // INTERNAL DATA

private:

  // problem information

  std::string           d_probTag;          // set in constructor, can be overwritten

  // status variables

  bool                  d_employScaling;    // call 'glp_scale_prob'
```

```
    bool                    d_employAdvBasis; // call 'glp_adv_basis'
    bool                    d_simplexPresol;  // set 'glp_smcp::presolve'
    bool                    d_integerPresol;  // set 'glp_iocp::presolve'

    ProblemStatus           d_probStatus;     // problem status
    ObjectiveSense          d_objSense;       // minimize or maximize
    ProblemKlass            d_probKlass;      // problem klass
    SolverType              d_solverType;     // solver type, either advisory or actual
    SolutionStatus          d_solnStatus;
    int                     d_solverRet;      // if set, GLPK three-digit integer, else 0
    int                     d_solnRet;        // if set, GLPK three-digit integer, else 0

    // problem administration information

    ReportingLevel          d_noise;          // also dependent on output suppression
    int                     d_solverCalls;    // solver call count
    int                     d_alertCount;     // local alert count

    // sparse matrix vectors used during problem construction
    // (naming as per GLPK tutorial)
    //
    // Josuttis (1999 p155) writes: "whenever you need an array
    // of type T for any reason (such as for an existing C
    // library) you can use a std::vector<T> and pass it the
    // address of the first element".  Namely &vec[0].  The
    // reason this works is that std::vector elements, like C
    // array elements, are required to be held in contiguous
    // memory.

    std::vector<int>        d_ia;             // constraint coefficient i index
    std::vector<int>        d_ja;             // constraint coefficient j index
    std::vector<double>     d_ar;             // constraint coefficient value

    // GLPK problem instance (a typedef to a C struct)

    glp_prob*               d_prob;           // initialized GLPK problem instance

    glp_smcp                d_parmSimplex;    // simplex parameters
    glp_iptcp               d_parmInterior;   // interior parameters
    glp_iocp                d_parmInteger;    // integer parameters

    // ------------------------------------------------------------
    // STATIC DATA

    // CAUTION: 's_os' is a stream pointer and not a stream reference

    static std::ostream*    s_os;             // combined class native and GLPK output
    static bool             s_noise;          // 'true' means output enabled

    static int              s_instanceCount;  // instance count, used for warnings only
    static const int        s_glpkMargin = 1; // GLPK does not often utilize array index 0
    static logga::spLogger s_logger;          // shared_ptr to single logger object

    // formatted output, for use in std::setw() and std::setprecision()

    static const int s_indexPad = 2;          // 0 to disable, 2 or more to zero-pad

    static const int s_W0 =  3;               // indent
    static const int s_W1 = 32;               // description
    static const int s_W2 =  2;               // ":"
    static const int s_W3 =  7;               // number alignment
    static const int s_P1 =  4;               // significant figures

  }; // class 'svif::SolverIf'

} // namespace svif

#endif // _SIGLP_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : context.h
//  file-create-date : Wed 06-May-2009 21:11 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : abstract context entity / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/context.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _CONTEXT_H_
#define _CONTEXT_H_

//  AD-HOC NOTES

//  LOCAL AND SYSTEM INCLUDES

#include "../b/entity.h"      // entity base class plus lazy linking

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

//  CODE

// ---------------------------------------------------------
//  CLASS           : Context (abstract proto-context class)
// ---------------------------------------------------------
//  Description  : abstract class for context entities
//  Role         : step in the inheritance web
//  Techniques   : pure virtual destructor
//  Status       : more-or-less complete
//
//  Design notes
//
//      This class provides for the following abstract classes:
//
//          - 'AmbientConditions'
//          - 'EconomicContext'
//          - 'PublicPolicyContext'
//
// ---------------------------------------------------------

class Context :
```

```
  public FullEntity
{
  // DISABLED

private:

  Context();                                  // zero-argument constructor
  Context(const Context& orig);               // copy constructor
  Context& operator= (const Context& orig);  // copy assignment operator

  // CREATORS

public:

  explicit
  Context(const std::string entityId) :
    FullEntity(entityId)
  { }

  explicit
  Context
  (const std::string entityId,
   Record&           record);

  virtual
  ~Context() = 0;                             // create abstract class

  // INTERNAL DATA

protected:

};

// ----------------------------------------------------------
//  CLASS           : AmbientConditions
// ----------------------------------------------------------

class AmbientConditions :
  public Context
{
  // DISABLED

private:

  AmbientConditions(const AmbientConditions& orig);              // copy constructor
  AmbientConditions& operator= (const AmbientConditions& orig);  // copy assignment oper

  // CREATORS

public:

  explicit
  AmbientConditions(const std::string entityId) :
    Context(entityId)
  { }

  explicit
  AmbientConditions
  (const std::string entityId,
   Record&           record);

  virtual
  ~AmbientConditions() = 0;                  // create abstract class

};

// ----------------------------------------------------------
//  CLASS           : EconomicContext
// ----------------------------------------------------------

class EconomicContext :
  public Context
{
  // DISABLED

private:

  EconomicContext();                                          // zero-argument ctor
  EconomicContext(const EconomicContext& orig);               // copy constructor
  EconomicContext& operator= (const EconomicContext& orig);   // copy assignment oper
```

```
  // CREATORS

public:

  explicit
  EconomicContext
  (const std::string entityId,
   Record&           record);

  virtual
  ~EconomicContext() = 0;                    // create abstract class

};

// ----------------------------------------------------------
//  CLASS           : PublicPolicyContext
// ----------------------------------------------------------

class PublicPolicyContext :
  public Context
{
  // DISABLED

private:

  PublicPolicyContext();                                          // zero-argument ctor
  PublicPolicyContext(const PublicPolicyContext& orig);           // copy constructor
  PublicPolicyContext& operator= (const PublicPolicyContext& orig); // copy assignment ope

  // CREATORS

public:

  explicit
  PublicPolicyContext
  (const std::string entityId,
   Record&           record);

  virtual
  ~PublicPolicyContext() = 0;                // create abstract class

};

#endif // _CONTEXT_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxamb01.h
//  file-create-date : Thu 07-May-2009 08:15 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete ambient conditions contexts 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxamb01.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  The contexts here cover ambient air.

//  HEADER GUARD

#ifndef _CXAMB01_H_
#define _CXAMB01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../e/context.h"     // abstract context entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

#include <boost/random.hpp>                       // convenience header for Boost.Random
#include <boost/math/distributions/rayleigh.hpp>  // Rayleigh statistical distribution

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

//  CODE

// ----------------------------------------------------------
//  CLASS (FUNCTOR) : xeona::SynWindMaker
// ----------------------------------------------------------
//  Description  : makes synthetic wind speed data
//  Role         : support for ambient air contexts based on internal simulation
//  Techniques   : functor, Boost.Math library, Boost.Random library
//  Status       : complete (but see caveats, particularly regarding volatility)
// ----------------------------------------------------------

namespace xeona
{
  class SynWindMaker
  {
    // DISABLED
```

```
  private:

    SynWindMaker();                                          // zero-argument constructor
    SynWindMaker& operator= (const SynWindMaker& orig);      // copy assignment operator

    // CREATORS

  public:

    SynWindMaker
    (const double mean);

    ~SynWindMaker();

    // MANIPULATORS

  public:

    double                                    // wind value in [m/s]
    operator()() const
      throw(std::domain_error, std::overflow_error);   // exception specification

    // INSTANCE DATA

  private:

    mutable int                                          d_callCount;  // call count [1]
    const double                                         d_mean;       // supplied mean
    const boost::math::rayleigh_distribution<double>     d_raydis;     // distribution

    // [1] necessary because operator() is 'const'

    // STATIC DATA

  private:

    static bool                           s_firstCall; // first call status
    static boost::mt19937                 s_engine;    // random generator engine
    static boost::uniform_real<double>    s_unidis;    // distribution
    static boost::variate_generator
    <
      boost::mt19937,
      boost::uniform_real<double>
    >                                     s_rng;       // later glue above for generator

    static logga::spLogger                s_logger;    // shared_ptr to single logger obj
  };

} // namespace 'xeona'

// ----------------------------------------------------------
//  CLASS           : CxAmbientAir
// ----------------------------------------------------------
//  Description  : interface base for ambient air contexts
//  Role         : step in the inheritance web
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

class CxAmbientAir :
  public AmbientConditions
{
  // DISABLED

private:

  CxAmbientAir();                                        // zero-argument constructor
  CxAmbientAir(const CxAmbientAir& orig);                // copy constructor
  CxAmbientAir& operator= (const CxAmbientAir& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  CxAmbientAir
  (const std::string entityId) :
    AmbientConditions(entityId)
  { }

  explicit
```

```
  CxAmbientAir
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxAmbientAir();

  // LINKING CALL

private:                                    // CAUTION: 'private' is correct

  virtual
  bool
  polylink(assign_ptr<Entity>& pass)        // CAUTION: 'Entity' is correct
  {
    s_logger->repx(logga::adhc, "entering virtual member function", "CxAmbientAir");
    const bool okay = pass.revamp(retFull<CxAmbientAir>());      // key call

    // low priority development reporting
    shared_ptr<CxAmbientAir> full = retFull<CxAmbientAir>();
    s_logger->repx(logga::adhc, "development reporting only", "");
    std::ostringstream put;
    put << "  full identifier : " << full->getIdentifier() << "\n"
        << "  full resource   : " << full.get()            << "\n";
    s_logger->putx(logga::adhc, put);

    return okay;                            // typically return directly from 'revamp'
  }

  // ACCESSORS

public:

  virtual
  double                                    // units [m/s]
  getWindSpeed
  (const int step) const  { return s_doubleZero; }

  virtual
  double
  getWindSpeedPeek
  (const int step,
   const int ahead) const { return s_doubleZero; }

  virtual
  double                                    // units [C]
  getAirTemp
  (const int step) const  { return s_doubleZero; }

  virtual
  double
  getAirTempPeek
  (const int step,
   const int ahead) const { return s_doubleZero; }

};

// ----------------------------------------------------------
//  CLASS           : CxAmbientAirTs
// ----------------------------------------------------------
//  Description  : ambient air context based on exogenous timeseries
//  Role         : ambient conditions support
//  Techniques   : std::vector
//  Status       : complete
// ----------------------------------------------------------

class CxAmbientAirTs :
  public CxAmbientAir
{
  // DISABLED

  CxAmbientAirTs();                                          // zero-argument constructor
  CxAmbientAirTs(const CxAmbientAirTs& orig);                // copy constructor
  CxAmbientAirTs& operator= (const CxAmbientAirTs& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  CxAmbientAirTs
```

```
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxAmbientAirTs();

  // ACCESSORS

public:

  virtual
  double                                    // units [m/s]
  getWindSpeed
  (const int step) const;

  virtual
  double
  getWindSpeedPeek
  (const int step,
   const int ahead = 1) const;              // steps look ahead

  virtual
  double                                    // units [C]
  getAirTemp
  (const int step) const;

  virtual
  double
  getAirTempPeek
  (const int step,
   const int ahead = 1) const;              // steps look ahead

  // INSTANCE DATA

private:

  const shared_ptr<std::vector<double> >    d_windSpeeds;
  const shared_ptr<std::vector<double> >    d_airTemps;

};

//  ==== XEDOC ==================================================
//
//  entity.cx-ambient-air-ts-0
//
//      class                                 > CxAmbientAirTs
//
//        ambient air context based on exogenous timeseries
//
//      builtin-remark s                      <
//
//      wind-speeds [m/s] F                   > 13.0 14.0 ..
//      air-temps [C] F                       > 10.0 11.0 ..
//
//  ============================================================

// ------------------------------------------------------------
//  CLASS           : CxAmbientAirSim
// ------------------------------------------------------------
//  Description  : ambient air context based on internal simulation
//  Role         : ambient conditions support
//  Techniques   : weather simulation
//  Status       : incomplete
// ------------------------------------------------------------

class CxAmbientAirSim :
  public CxAmbientAir
{
  // DISABLED

  CxAmbientAirSim();                                        // zero-argument constructor
  CxAmbientAirSim(const CxAmbientAirSim& orig);             // copy constructor
  CxAmbientAirSim& operator= (const CxAmbientAirSim& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  CxAmbientAirSim
  (const std::string entityId,
```

```
   Record&           record);

  virtual
  ~CxAmbientAirSim();

  // ACCESSORS

public:

  virtual
  double                                    // units [m/s]
  getWindSpeed
  (const int step) const;

  virtual
  double
  getWindSpeedPeek
  (const int step,
   const int ahead = 1) const;              // steps look ahead

  virtual
  double                                    // units [C]
  getAirTemp
  (const int step) const;

  virtual
  double
  getAirTempPeek
  (const int step,
   const int ahead = 1) const;              // steps look ahead

  // INSTANCE DATA

private:

  const double&                       d_meanWindSpeed;
  const double&                       d_constantAirTemp;    // constant
  shared_ptr<std::vector<double> >    d_windSpeeds;         // generated timeseries

};

//  ==== XEDOC ==================================================
//
//  entity.cx-ambient-air-sim-0
//
//      class                                   > CxAmbientAirSim
//
//        an ambient air context with wind speeds sampled from a
//        stateless Rayleigh distribution -- meaning that the
//        prior values do not influence the current values
//
//      builtin-remark s                        <
//
//      mean-wind-speed [m/s] f                 > 10.5
//      constant-air-temp [C] f                 > 15.0
//
//        mean-wind-speed is measured at 10m above ground
//
//        constant-air-temp allows this entity to provide
//        temperature data in line with the CxAmbientAir
//        interface
//
//      wind-speeds [m/s] F                     < 0.0 ..
//
//        the wind-speeds are duly calculated
//
//  ============================================================

#endif // _CXAMB01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxamb02.h
//  file-create-date : Thu 07-May-2009 13:52 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete ambient conditions contexts 2 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxamb02.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  The contexts here cover ambient solar.

//  HEADER GUARD

#ifndef _CXAMB02_H_
#define _CXAMB02_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../e/context.h"     // abstract context entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>                 // n-tuples, ref(), cref()

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                // constructor argument

//  CODE

// ----------------------------------------------------------
//  CLASS           : CxAmbientSolar (abstract)
// ----------------------------------------------------------
//  Description  : interface base for ambient solar contexts
//  Role         : step in the inheritance web
//  Techniques   : pure virtual functions
//  Status       : complete
// ----------------------------------------------------------

class CxAmbientSolar :
  public AmbientConditions
{
  // DISABLED

private:
```

```
  CxAmbientSolar();                                          // zero-argument constructor
  CxAmbientSolar(const CxAmbientSolar& orig);                // copy constructor
  CxAmbientSolar& operator= (const CxAmbientSolar& orig);    // copy assignment operator

  // CREATORS

public:

  explicit
  CxAmbientSolar
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxAmbientSolar() = 0;                     // create abstract class

  // ACCESSORS

public:

  virtual
  boost::tuple
  <double,                                   // direct component, units [W/m^2]
   double>                                   // diffuse component, units [W/m^2]
  getSolar
  (const int step) const = 0;

  virtual
  boost::tuple
  <double,
   double>
  getSolarPeek
  (const int step,
   const int ahead) const = 0;

};

// ----------------------------------------------------------
//  CLASS           : CxAmbientSolarTs
// ----------------------------------------------------------
//  Description  : ambient solar context based on exogenous timeseries
//  Role         : ambient conditions support
//  Techniques   : std::vector
//  Status       : complete
// ----------------------------------------------------------

class CxAmbientSolarTs :
  public CxAmbientSolar
{
  // DISABLED

  CxAmbientSolarTs();                                          // zero-argument constructor
  CxAmbientSolarTs(const CxAmbientSolarTs& orig);              // copy constructor
  CxAmbientSolarTs& operator= (const CxAmbientSolarTs& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  CxAmbientSolarTs
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxAmbientSolarTs();

  // ACCESSORS

public:

  virtual
  boost::tuple
  <double,                                   // direct component, units [W/m^2]
   double>                                   // diffuse component, units [W/m^2]
  getSolar
  (const int step) const;

  virtual
  boost::tuple
  <double,
```

```
   double>
  getSolarPeek
  (const int step,
   const int ahead = 1) const;

  // INSTANCE DATA

private:

  const shared_ptr<std::vector<double> >    d_solarDirect;
  const shared_ptr<std::vector<double> >    d_solarDifuse;

};

//  ==== XEDOC ==================================================
//
//  entity.cx-ambient-solar-ts-0
//
//      class                                   > CxAmbientSolarTs
//
//        ambient solar context based on exogenous timeseries
//
//      builtin-remark s                        <
//
//      solar-direct [W/m^2] F                  > 500.0 ..
//      solar-diffuse [W/m^2] F                 > 500.0 ..
//
//        typical average combined value is 324W/m^2
//
//  ============================================================

#endif // _CXAMB02_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxecon01.h
//  file-create-date : Thu 07-May-2009 08:16 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete economic contexts 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxecon01.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _CXECON01_H_
#define _CXECON01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../e/context.h"     // abstract context entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

//  CODE

// ----------------------------------------------------------
//  CLASS           : CxCommercial (abstract)
// ----------------------------------------------------------
//  Description  : interface base for commercial contexts
//  Role         : step in the inheritance web
//  Techniques   : pure virtual functions
//  Status       : incomplete
// ----------------------------------------------------------

class CxCommercial :
  public EconomicContext
{
  // DISABLED

private:

  CxCommercial();                                     // zero-argument constructor
  CxCommercial(const CxCommercial& orig);             // copy constructor
  CxCommercial& operator= (const CxCommercial& orig);  // copy assignment operator

  // CREATORS
```

```
public:

  explicit
  CxCommercial
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxCommercial() = 0;                      // create abstract class

  // ACCESSORS

public:

  virtual
  double                                    // decimal form unit [-/y], say 0.2
  getComInterestRate() const = 0;

};

// ----------------------------------------------------------
//  CLASS           : CxCommercialFix
// ----------------------------------------------------------

class CxCommercialFix :
  public CxCommercial
{
  // DISABLED

  CxCommercialFix();                                          // zero-argument constructor
  CxCommercialFix(const CxCommercialFix& orig);               // copy constructor
  CxCommercialFix& operator= (const CxCommercialFix& orig); // copy assignment operator

  // CREATORS

public:

  explicit
  CxCommercialFix
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxCommercialFix();

  // ACCESSORS

public:

  virtual
  double                                    // decimal form unit [-/y], say 0.2
  getComInterestRate() const;

  // INSTANCE DATA

private:

  const double&    d_comInterestRate;

};

//  ==== XEDOC ================================================
//
//  entity.cx-commercial-fix-0
//
//      class                                   > CxCommercialFix
//
//        commercial context using fixed data
//
//      builtin-remark s                        <
//
//      commercial-interest-rate [-/y] f        > 0.2
//
//  ===========================================================

#endif // _CXECON01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxpubpol01.h
//  file-create-date : Thu 07-May-2009 08:27 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete public policy contexts 1 / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxpol01.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  The contexts here cover public policy measures.

//  HEADER GUARD

#ifndef _CXPUBPOL01_H_
#define _CXPUBPOL01_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../e/context.h"     // abstract context entity

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // constructor argument

//  CODE

// ----------------------------------------------------------
//  CLASS           : CxPublicPolicyA
// ----------------------------------------------------------

class CxPublicPolicyA :
  public PublicPolicyContext
{
  // DISABLED

private:

  CxPublicPolicyA();                                        // zero-argument constructor
  CxPublicPolicyA(const CxPublicPolicyA& orig);             // copy constructor
  CxPublicPolicyA& operator= (const CxPublicPolicyA& orig); // copy assignment operator

  // CREATORS

public:
```

```
  explicit
  CxPublicPolicyA
  (const std::string entityId,
   Record&           record);

  virtual
  ~CxPublicPolicyA() = 0;                    // create abstract class

};

//  ==== xedoc ======================================================
//
//  entity.cx-public-policy-a-0
//
//      class                                    > CxPublicPolicyA
//
//        public policy context
//
//      builtin-remark s                         <
//
//  ==================================================================

#endif // _CXPUBPOL01_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cta.h
//  file-create-date : Fri 06-Feb-2009 15:11 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : captrans algorithm / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/cta.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  The "captrans" algorithm is described in detail in my (Robbie
//  Morrison) PhD thesis.

//  HEADER GUARD

#ifndef _CTA_H_
#define _CTA_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  FORWARD (PARTIAL) DECLARATIONS

class DomainController;

//  CODE

// ----------------------------------------------------------
//  CLASS           : CapTransAlg (abstract base class)
// ----------------------------------------------------------
//  Description  : base class for various "captrans" algorithms (CTA)
//  Role         : traverse domain graph and make 'capset' and 'transolve' calls
//  Techniques   : pure virtual, re-invoked by 'Overseer' each interval
//  Status       : complete (referring to this base class)
//
//  Design notes
//
//      Capacity mode
```

```
//
//          The 'capacityMode' is (depending on the derived class
//          implementation) taken by the function 'captrans',
//          range checked, and then passed on to the 'capset' and
//          'transolve' calls as these occur -- hence, the
//          'capacityMode' value has no direct influence on
//          course of the "captrans" algorithm.
//
//      Algorithm description
//
//          The CTA (including its simple and hop-relitate
//          variants) is NOT described here.  Refer instead to my
//          (Robbie Morrison) PhD thesis for more information.
//          The CTA is relatively complicated!
//
// ----------------------------------------------------------

class CapTransAlg
{
  // DISABLED

private:

  CapTransAlg();                                          // zero-argument ctor
  CapTransAlg(const CapTransAlg& orig);                   // copy constructor
  CapTransAlg& operator= (const CapTransAlg& orig);       // copy assignment

  // CREATORS

public:

  CapTransAlg
  (const int                                              step,       // current step
   const std::vector<shared_ptr<DomainController> >& domcons);   // ranked

  virtual
  ~CapTransAlg();

  // MANIPULATORS

  virtual
  tribool                                       // 'indeterminate' if no originating domains
  captrans
  (const xeona::DomainMode capacityMode)
    = 0;

  // INSTANCE DATA

protected:

  const int                                               d_step;
  const std::vector<shared_ptr<DomainController> >        d_originatingDomcons;

  // STATIC DATA

protected:

  static logga::spLogger    s_logger;        // shared_ptr to single logger object

};

// ----------------------------------------------------------
//  CLASS           : CtaFixed
// ----------------------------------------------------------
//  Description  : stub for static capacity setting
//  Role         : <none>
//  Techniques   : <none>
//  Status       : hollow, reports noisily if construction attempted
//
//  CAUTION: development status
//
//      This class is currently hollow -- and will likely stay
//      that way for quite some time.
//
// ----------------------------------------------------------

class CtaFixed :
  public CapTransAlg
{
public:
```

```
  CtaFixed
  (const int                                          step,
   const std::vector<shared_ptr<DomainController> >& domcons);

  tribool                                     // 'indeterminate' if no originating domains
  captrans
  (const xeona::DomainMode capacityMode);

};

// ----------------------------------------------------------
//  CLASS           : CtaSimple
// ----------------------------------------------------------
//  Description  : derived class
//  Role         : see 'CapTransAlg'
//  Techniques   : see 'CtaSimple::captrans'
//  Status       : see 'CtaSimple::captrans'
//
//  CAUTION: dual-implementation (temporary measure)
//
//      This class currently contains code for both the "simple"
//      and "hop-relit" variants selected by way of the 'XE_CTA'
//      cpp macro.
//
// ----------------------------------------------------------

class CtaSimple :
  public CapTransAlg
{
public:

  CtaSimple
  (const int                                          step,
   const std::vector<shared_ptr<DomainController> >& domcons);

  ~CtaSimple();

  tribool                                     // 'indeterminate' if no originating domains
  captrans
  (const xeona::DomainMode capacityMode);

};

// ----------------------------------------------------------
//  CLASS           : CtaHopRelit
// ----------------------------------------------------------

// CAUTION: this class contains private creators for the time
// being, but should re-instigate later to allow XEM-level
// selection between 'simple' and 'hop-relit' variants -- note
// also there is no allied code in the implementation file at
// present

class CtaHopRelit :
  public CapTransAlg
{
private:                                      // fully disabled for now
  CtaHopRelit();                                        // zero-argument ctor
  CtaHopRelit(const CtaHopRelit& orig);                 // copy constructor
  CtaHopRelit& operator= (const CtaHopRelit& orig);     // copy assignment
  ~CtaHopRelit();                                       // destructor
};

#endif // _CTA_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : gatesreg.h
//  file-create-date : Sat 20-Feb-2010 19:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : find and register gateways / header
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/gatesreg.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _GATESREG_H_
#define _GATESREG_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ---------------------------------------------------------
//  FREE FUNCTION   : xeona::registerGates
// ---------------------------------------------------------
//  Description  : finalize reciprocal information for gateway and domain controllers
//  Role         : point of contact for registering gateways, called by 'xeona::simulate'
//  Note         : one pass protection
//  Status       : complete
// ---------------------------------------------------------

namespace xeona
{
  bool                                      // 'true' means ran to completion
  registerGates();
}

#endif // _GATESREG_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : ospinfo.h
//  file-create-date : Mon 19-Oct-2009 11:04 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain mode interpretation / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/ospinfo.h $
//
//  GENERAL NOTES FOR THIS FILE

//  HEADER GUARD

#ifndef _OSPINFO_H_
#define _OSPINFO_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../f/ospmodes.h"    // domain mode enums (header only)

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

//  CODE

// ------------------------------------------------------------
//  CLASS           : DomainModeInfo
// ------------------------------------------------------------

class DomainModeInfo
{
  // FRIENDS

  friend class DomainModeDatabase;

  // CREATORS

private:

  DomainModeInfo
  (const xeona::DomainMode mode,
   const std::string       shortMsg,
   const std::string       longMsg);

  // ACCESSORS

private:

  std::string
```

```
  shortform() const;

  std::string
  longform() const;

  // INSTANCE DATA

private:

  xeona::DomainMode    d_mode;              // mode
  std::string          d_shortMsg;          // based on the enum identifier
  std::string          d_longMsg;           // long-based message
  std::string          d_modeKind;          // set here

};

// ----------------------------------------------------------
//  CLASS           : DomainModeDatabase
// ----------------------------------------------------------

class DomainModeDatabase
{
public:

  DomainModeDatabase();

  std::vector<std::string>
  longform
  (const int domainModes) const;

  std::string
  getInfo
  (const int domainMode) const;

  void
  test
  (std::ostringstream& oss) const;

  // INTERNAL DATA

private:

  std::vector<DomainModeInfo>    d_infos;    // database
  static logga::spLogger         s_logger;   // shared_ptr to single logger object

};

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::infoDomainModeLong (twice)
// ----------------------------------------------------------
//  Description  : interpret 'xeona::DomainMode' or integer-equivalent values
//  Role         : use in logging messages
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Usage
//
//      The following example assumes that the domain mode value
//      captures just one domain mode.
//
//          std::vector<std::string> modes = xeona::infoDomainModeLong(d_commitmentMode);
//          std::string mode = "(some problem)";           // default value
//          if ( modes.size() == 1 ) mode = modes.front(); // unique mode encountered
//
// ----------------------------------------------------------

namespace xeona
{
  std::vector<std::string>
  infoDomainModeLong
  (const int domainModes);                  // can be non-pure

  std::string
  infoDomainModeLong
  (const int domainModes,                   // can be non-pure
   const int padding);

} // namespace 'xeona'

#endif // _OSPINFO_H_
```

```
//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : trav.h
//  file-create-date : Wed 11-Feb-2009 13:42 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain graph traversal class / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/trav.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  The code in this file is based on the unit 'cta' and in
//  particular code in the CTA reset procedure.

#ifndef _TRAV_H_
#define _TRAV_H_

//  LOCAL AND SYSTEM INCLUDES

#include "../a/logger_fwd.h"  // some forward declarations from "logger.h"
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers

#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/tuple/tuple.hpp>                 // n-tuples, ref(), cref()

//  FORWARD (PARTIAL) DECLARATIONS

class DomainController;
class CostSet;

//  TYPEDEFS

typedef void (DomainController::* funcPtrVoid)();
typedef void (DomainController::* funcPtrStep)(const int);
typedef void (DomainController::* funcPtrStepCs3)
  (const int, CostSet&, CostSet&, CostSet&);

// The above are pointer-to-member function types, see Stroustrup
// (1997 p419), Stephens etal (2006 pp539-541), Loudon (2003
// pp26-27).  This typedef makes the subsequent syntax easier.

//  CODE

// ---------------------------------------------------------
//  CLASS           : TravDepthFirst
// ---------------------------------------------------------
//  Description  : depth first domain graph traversal, used to call passed-in functions
//  Role         : domain (and thereby selgate) visitation invoked by 'Overseer' singleton
//  Techniques   : depth first search, pointer-to-member function argument (not pretty)
//  Status       : complete
//
```

```
//  Design notes
//
//      The 'TravDepthFirst::call' calling syntax needs to be
//      split into TWO statements, as indicated in lines 2 and 3
//      below:
//
//          shared_ptr<TravDepthFirst> dfs(new TravDepthFirst(rankedOrigDomains));
//          funcPtr f = &DomainController::function;
//          const int fCount = dfs->callDomains(f, step);
//
//      The underlying function argument needs to be of the
//      following general type:
//
//          void DomainController::someFunc(const int)
//
//      A graph component is a maximally connected subgraph --
//      thus, in the case of 'xeona', an energy island.
//
// ------------------------------------------------------------

class TravDepthFirst
{
  // DISABLED

private:

  TravDepthFirst();                                          // zero-argument ctor
  TravDepthFirst(const TravDepthFirst& orig);                // copy constructor
  TravDepthFirst& operator= (const TravDepthFirst& orig);    // copy assignment

  // CREATORS

public:

  TravDepthFirst                                                    // calls 'depthFirstTrav'
  (const std::vector<shared_ptr<DomainController> >& domcons);   // ranked domains

  ~TravDepthFirst();

  // ACCESSORS

  int                                       // number of originating domains
  getOrigDomconsCount() const;

  int                                       // total number of domains
  getAllDomconsCount() const;

  int                                       // number of maximally connected sub-graphs
  getComponentsCount() const;

  bool                                      // 'true' indicates success
  getAllDomcons                             // all domains in topological sort order
  (std::vector<shared_ptr<DomainController> >& allDomcons) const;

  // MANIPULATORS - note overloading

  const int                                 // number of visited domains
  callDomains                               // interval-specific call
  (funcPtrStep func,                        // typedef, argument type must match
   const int   step);                       // step passed thru

  const int                                 // number of visited domains
  callDomains                               // start and end of run call
  (funcPtrVoid func);                       // typedef, argument type must match

  const int
  callDomains
  (funcPtrStepCs3 func,
   const int      step,
   CostSet&       var,
   CostSet&       fix,
   CostSet&       emb);

  // UTILITY FUNCTIONS

private:

  bool                                      // 'true' indicates success
  depthFirstTrav();                         // workhorse algorithm

  // INSTANCE DATA
```

```
private:

  const std::vector<shared_ptr<DomainController> >    d_originatingDomcons;

  std::vector<shared_ptr<DomainController> >          d_allDomcons;
  int                                                 d_componentCount;

  // STATIC DATA

protected:

  static logga::spLogger    s_logger;        // shared_ptr to single logger object

};

#endif // _TRAV_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : logger_fwd.h
//  file-create-date : Mon 16-Jul-2007 19:30 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : subset of "logger.h" (avoid hash-including entire header) / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/logger_fwd.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This file is used to avoid hash-including the entire logging unit header.

//  HEADER GUARD

#ifndef _LOGGER_FWD_H_
#define _LOGGER_FWD_H_
#ifndef _LOGGER_H_            // skip if "../a/logger.h"

//  LOCAL AND SYSTEM INCLUDES

#include "../c/smart_ptr.h"   // switch easily between TR1 and Boost smart pointers

//  FORWARD (PARTIAL) DECLARATIONS

namespace logga               // forward declaration for logger object
{
  class Logger;
  typedef shared_ptr<Logger> spLogger;
}

#endif // _LOGGER_H_
#endif // _LOGGER_FWD_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : license.h
//  file-create-date : Thu 08-Nov-2007 18:39 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : GNU GPLv3 software license / header
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/license.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  A straight hash-include into 'appinfo.cc'.

//  HEADER GUARD

//  #ifndef _LICENSE_H_
//  #define _LICENSE_H_

"                    GNU GENERAL PUBLIC LICENSE\n"
"                       Version 3, 29 June 2007\n"
"\n"
"   Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>\n"
"   Everyone is permitted to copy and distribute verbatim copies\n"
"   of this license document, but changing it is not allowed.\n"
"\n"
"                            Preamble\n"
"\n"
"    The GNU General Public License is a free, copyleft license for\n"
"  software and other kinds of works.\n"
"\n"
"    The licenses for most software and other practical works are designed\n"
"  to take away your freedom to share and change the works.  By contrast,\n"
"  the GNU General Public License is intended to guarantee your freedom to\n"
"  share and change all versions of a program--to make sure it remains free\n"
"  software for all its users.  We, the Free Software Foundation, use the\n"
"  GNU General Public License for most of our software; it applies also to\n"
"  any other work released this way by its authors.  You can apply it to\n"
"  your programs, too.\n"
"\n"
"    When we speak of free software, we are referring to freedom, not\n"
"  price.  Our General Public Licenses are designed to make sure that you\n"
"  have the freedom to distribute copies of free software (and charge for\n"
"  them if you wish), that you receive source code or can get it if you\n"
"  want it, that you can change the software or use pieces of it in new\n"
"  free programs, and that you know you can do these things.\n"
"\n"
"    To protect your rights, we need to prevent others from denying you\n"
"  these rights or asking you to surrender the rights.  Therefore, you have\n"
"  certain responsibilities if you distribute copies of the software, or if\n"
"  you modify it: responsibilities to respect the freedom of others.\n"
"\n"
"    For example, if you distribute copies of such a program, whether\n"
"  gratis or for a fee, you must pass on to the recipients the same\n"
```

```
"  freedoms that you received.  You must make sure that they, too, receive\n"
"  or can get the source code.  And you must show them these terms so they\n"
"  know their rights.\n"
"\n"
"    Developers that use the GNU GPL protect your rights with two steps:\n"
"  (1) assert copyright on the software, and (2) offer you this License\n"
"  giving you legal permission to copy, distribute and/or modify it.\n"
"\n"
"    For the developers' and authors' protection, the GPL clearly explains\n"
"  that there is no warranty for this free software.  For both users' and\n"
"  authors' sake, the GPL requires that modified versions be marked as\n"
"  changed, so that their problems will not be attributed erroneously to\n"
"  authors of previous versions.\n"
"\n"
"    Some devices are designed to deny users access to install or run\n"
"  modified versions of the software inside them, although the manufacturer\n"
"  can do so.  This is fundamentally incompatible with the aim of\n"
"  protecting users' freedom to change the software.  The systematic\n"
"  pattern of such abuse occurs in the area of products for individuals to\n"
"  use, which is precisely where it is most unacceptable.  Therefore, we\n"
"  have designed this version of the GPL to prohibit the practice for those\n"
"  products.  If such problems arise substantially in other domains, we\n"
"  stand ready to extend this provision to those domains in future versions\n"
"  of the GPL, as needed to protect the freedom of users.\n"
"\n"
"    Finally, every program is threatened constantly by software patents.\n"
"  States should not allow patents to restrict development and use of\n"
"  software on general-purpose computers, but in those that do, we wish to\n"
"  avoid the special danger that patents applied to a free program could\n"
"  make it effectively proprietary.  To prevent this, the GPL assures that\n"
"  patents cannot be used to render the program non-free.\n"
"\n"
"    The precise terms and conditions for copying, distribution and\n"
"  modification follow.\n"
"\n"
"                         TERMS AND CONDITIONS\n"
"\n"
"    0. Definitions.\n"
"\n"
"    \"This License\" refers to version 3 of the GNU General Public License.\n"
"\n"
"    \"Copyright\" also means copyright-like laws that apply to other kinds of\n"
"  works, such as semiconductor masks.\n"
"\n"
"    \"The Program\" refers to any copyrightable work licensed under this\n"
"  License.  Each licensee is addressed as \"you\".  \"Licensees\" and\n"
"  \"recipients\" may be individuals or organizations.\n"
"\n"
"    To \"modify\" a work means to copy from or adapt all or part of the work\n"
"  in a fashion requiring copyright permission, other than the making of an\n"
"  exact copy.  The resulting work is called a \"modified version\" of the\n"
"  earlier work or a work \"based on\" the earlier work.\n"
"\n"
"    A \"covered work\" means either the unmodified Program or a work based\n"
"  on the Program.\n"
"\n"
"    To \"propagate\" a work means to do anything with it that, without\n"
"  permission, would make you directly or secondarily liable for\n"
"  infringement under applicable copyright law, except executing it on a\n"
"  computer or modifying a private copy.  Propagation includes copying,\n"
"  distribution (with or without modification), making available to the\n"
"  public, and in some countries other activities as well.\n"
"\n"
"    To \"convey\" a work means any kind of propagation that enables other\n"
"  parties to make or receive copies.  Mere interaction with a user through\n"
"  a computer network, with no transfer of a copy, is not conveying.\n"
"\n"
"    An interactive user interface displays \"Appropriate Legal Notices\"\n"
"  to the extent that it includes a convenient and prominently visible\n"
"  feature that (1) displays an appropriate copyright notice, and (2)\n"
"  tells the user that there is no warranty for the work (except to the\n"
"  extent that warranties are provided), that licensees may convey the\n"
"  work under this License, and how to view a copy of this License.  If\n"
"  the interface presents a list of user commands or options, such as a\n"
"  menu, a prominent item in the list meets this criterion.\n"
"\n"
"    1. Source Code.\n"
"\n"
"    The \"source code\" for a work means the preferred form of the work\n"
"  for making modifications to it.  \"Object code\" means any non-source\n"
"  form of a work.\n"
```

```
"\n"
"    A \"Standard Interface\" means an interface that either is an official\n"
"  standard defined by a recognized standards body, or, in the case of\n"
"  interfaces specified for a particular programming language, one that\n"
"  is widely used among developers working in that language.\n"
"\n"
"    The \"System Libraries\" of an executable work include anything, other\n"
"  than the work as a whole, that (a) is included in the normal form of\n"
"  packaging a Major Component, but which is not part of that Major\n"
"  Component, and (b) serves only to enable use of the work with that\n"
"  Major Component, or to implement a Standard Interface for which an\n"
"  implementation is available to the public in source code form.  A\n"
"  \"Major Component\", in this context, means a major essential component\n"
"  (kernel, window system, and so on) of the specific operating system\n"
"  (if any) on which the executable work runs, or a compiler used to\n"
"  produce the work, or an object code interpreter used to run it.\n"
"\n"
"    The \"Corresponding Source\" for a work in object code form means all\n"
"  the source code needed to generate, install, and (for an executable\n"
"  work) run the object code and to modify the work, including scripts to\n"
"  control those activities.  However, it does not include the work's\n"
"  System Libraries, or general-purpose tools or generally available free\n"
"  programs which are used unmodified in performing those activities but\n"
"  which are not part of the work.  For example, Corresponding Source\n"
"  includes interface definition files associated with source files for\n"
"  the work, and the source code for shared libraries and dynamically\n"
"  linked subprograms that the work is specifically designed to require,\n"
"  such as by intimate data communication or control flow between those\n"
"  subprograms and other parts of the work.\n"
"\n"
"    The Corresponding Source need not include anything that users\n"
"  can regenerate automatically from other parts of the Corresponding\n"
"  Source.\n"
"\n"
"    The Corresponding Source for a work in source code form is that\n"
"  same work.\n"
"\n"
"    2. Basic Permissions.\n"
"\n"
"    All rights granted under this License are granted for the term of\n"
"  copyright on the Program, and are irrevocable provided the stated\n"
"  conditions are met.  This License explicitly affirms your unlimited\n"
"  permission to run the unmodified Program.  The output from running a\n"
"  covered work is covered by this License only if the output, given its\n"
"  content, constitutes a covered work.  This License acknowledges your\n"
"  rights of fair use or other equivalent, as provided by copyright law.\n"
"\n"
"    You may make, run and propagate covered works that you do not\n"
"  convey, without conditions so long as your license otherwise remains\n"
"  in force.  You may convey covered works to others for the sole purpose\n"
"  of having them make modifications exclusively for you, or provide you\n"
"  with facilities for running those works, provided that you comply with\n"
"  the terms of this License in conveying all material for which you do\n"
"  not control copyright.  Those thus making or running the covered works\n"
"  for you must do so exclusively on your behalf, under your direction\n"
"  and control, on terms that prohibit them from making any copies of\n"
"  your copyrighted material outside their relationship with you.\n"
"\n"
"    Conveying under any other circumstances is permitted solely under\n"
"  the conditions stated below.  Sublicensing is not allowed; section 10\n"
"  makes it unnecessary.\n"
"\n"
"    3. Protecting Users' Legal Rights From Anti-Circumvention Law.\n"
"\n"
"    No covered work shall be deemed part of an effective technological\n"
"  measure under any applicable law fulfilling obligations under article\n"
"  11 of the WIPO copyright treaty adopted on 20 December 1996, or\n"
"  similar laws prohibiting or restricting circumvention of such\n"
"  measures.\n"
"\n"
"    When you convey a covered work, you waive any legal power to forbid\n"
"  circumvention of technological measures to the extent such circumvention\n"
"  is effected by exercising rights under this License with respect to\n"
"  the covered work, and you disclaim any intention to limit operation or\n"
"  modification of the work as a means of enforcing, against the work's\n"
"  users, your or third parties' legal rights to forbid circumvention of\n"
"  technological measures.\n"
"\n"
"    4. Conveying Verbatim Copies.\n"
"\n"
"    You may convey verbatim copies of the Program's source code as you\n"
```

```
"  receive it, in any medium, provided that you conspicuously and\n"
"  appropriately publish on each copy an appropriate copyright notice;\n"
"  keep intact all notices stating that this License and any\n"
"  non-permissive terms added in accord with section 7 apply to the code;\n"
"  keep intact all notices of the absence of any warranty; and give all\n"
"  recipients a copy of this License along with the Program.\n"
"\n"
"    You may charge any price or no price for each copy that you convey,\n"
"  and you may offer support or warranty protection for a fee.\n"
"\n"
"    5. Conveying Modified Source Versions.\n"
"\n"
"    You may convey a work based on the Program, or the modifications to\n"
"  produce it from the Program, in the form of source code under the\n"
"  terms of section 4, provided that you also meet all of these conditions:\n"
"\n"
"      a) The work must carry prominent notices stating that you modified\n"
"      it, and giving a relevant date.\n"
"\n"
"      b) The work must carry prominent notices stating that it is\n"
"      released under this License and any conditions added under section\n"
"      7.  This requirement modifies the requirement in section 4 to\n"
"      \"keep intact all notices\".\n"
"\n"
"      c) You must license the entire work, as a whole, under this\n"
"      License to anyone who comes into possession of a copy.  This\n"
"      License will therefore apply, along with any applicable section 7\n"
"      additional terms, to the whole of the work, and all its parts,\n"
"      regardless of how they are packaged.  This License gives no\n"
"      permission to license the work in any other way, but it does not\n"
"      invalidate such permission if you have separately received it.\n"
"\n"
"      d) If the work has interactive user interfaces, each must display\n"
"      Appropriate Legal Notices; however, if the Program has interactive\n"
"      interfaces that do not display Appropriate Legal Notices, your\n"
"      work need not make them do so.\n"
"\n"
"    A compilation of a covered work with other separate and independent\n"
"  works, which are not by their nature extensions of the covered work,\n"
"  and which are not combined with it such as to form a larger program,\n"
"  in or on a volume of a storage or distribution medium, is called an\n"
"  \"aggregate\" if the compilation and its resulting copyright are not\n"
"  used to limit the access or legal rights of the compilation's users\n"
"  beyond what the individual works permit.  Inclusion of a covered work\n"
"  in an aggregate does not cause this License to apply to the other\n"
"  parts of the aggregate.\n"
"\n"
"    6. Conveying Non-Source Forms.\n"
"\n"
"    You may convey a covered work in object code form under the terms\n"
"  of sections 4 and 5, provided that you also convey the\n"
"  machine-readable Corresponding Source under the terms of this License,\n"
"  in one of these ways:\n"
"\n"
"      a) Convey the object code in, or embodied in, a physical product\n"
"      (including a physical distribution medium), accompanied by the\n"
"      Corresponding Source fixed on a durable physical medium\n"
"      customarily used for software interchange.\n"
"\n"
"      b) Convey the object code in, or embodied in, a physical product\n"
"      (including a physical distribution medium), accompanied by a\n"
"      written offer, valid for at least three years and valid for as\n"
"      long as you offer spare parts or customer support for that product\n"
"      model, to give anyone who possesses the object code either (1) a\n"
"      copy of the Corresponding Source for all the software in the\n"
"      product that is covered by this License, on a durable physical\n"
"      medium customarily used for software interchange, for a price no\n"
"      more than your reasonable cost of physically performing this\n"
"      conveying of source, or (2) access to copy the\n"
"      Corresponding Source from a network server at no charge.\n"
"\n"
"      c) Convey individual copies of the object code with a copy of the\n"
"      written offer to provide the Corresponding Source.  This\n"
"      alternative is allowed only occasionally and noncommercially, and\n"
"      only if you received the object code with such an offer, in accord\n"
"      with subsection 6b.\n"
"\n"
"      d) Convey the object code by offering access from a designated\n"
"      place (gratis or for a charge), and offer equivalent access to the\n"
"      Corresponding Source in the same way through the same place at no\n"
"      further charge.  You need not require recipients to copy the\n"
```

```
"      Corresponding Source along with the object code.  If the place to\n"
"      copy the object code is a network server, the Corresponding Source\n"
"      may be on a different server (operated by you or a third party)\n"
"      that supports equivalent copying facilities, provided you maintain\n"
"      clear directions next to the object code saying where to find the\n"
"      Corresponding Source.  Regardless of what server hosts the\n"
"      Corresponding Source, you remain obligated to ensure that it is\n"
"      available for as long as needed to satisfy these requirements.\n"
"\n"
"      e) Convey the object code using peer-to-peer transmission, provided\n"
"      you inform other peers where the object code and Corresponding\n"
"      Source of the work are being offered to the general public at no\n"
"      charge under subsection 6d.\n"
"\n"
"    A separable portion of the object code, whose source code is excluded\n"
"  from the Corresponding Source as a System Library, need not be\n"
"  included in conveying the object code work.\n"
"\n"
"    A \"User Product\" is either (1) a \"consumer product\", which means any\n"
"  tangible personal property which is normally used for personal, family,\n"
"  or household purposes, or (2) anything designed or sold for incorporation\n"
"  into a dwelling.  In determining whether a product is a consumer product,\n"
"  doubtful cases shall be resolved in favor of coverage.  For a particular\n"
"  product received by a particular user, \"normally used\" refers to a\n"
"  typical or common use of that class of product, regardless of the status\n"
"  of the particular user or of the way in which the particular user\n"
"  actually uses, or expects or is expected to use, the product.  A product\n"
"  is a consumer product regardless of whether the product has substantial\n"
"  commercial, industrial or non-consumer uses, unless such uses represent\n"
"  the only significant mode of use of the product.\n"
"\n"
"    \"Installation Information\" for a User Product means any methods,\n"
"  procedures, authorization keys, or other information required to install\n"
"  and execute modified versions of a covered work in that User Product from\n"
"  a modified version of its Corresponding Source.  The information must\n"
"  suffice to ensure that the continued functioning of the modified object\n"
"  code is in no case prevented or interfered with solely because\n"
"  modification has been made.\n"
"\n"
"    If you convey an object code work under this section in, or with, or\n"
"  specifically for use in, a User Product, and the conveying occurs as\n"
"  part of a transaction in which the right of possession and use of the\n"
"  User Product is transferred to the recipient in perpetuity or for a\n"
"  fixed term (regardless of how the transaction is characterized), the\n"
"  Corresponding Source conveyed under this section must be accompanied\n"
"  by the Installation Information.  But this requirement does not apply\n"
"  if neither you nor any third party retains the ability to install\n"
"  modified object code on the User Product (for example, the work has\n"
"  been installed in ROM).\n"
"\n"
"    The requirement to provide Installation Information does not include a\n"
"  requirement to continue to provide support service, warranty, or updates\n"
"  for a work that has been modified or installed by the recipient, or for\n"
"  the User Product in which it has been modified or installed.  Access to a\n"
"  network may be denied when the modification itself materially and\n"
"  adversely affects the operation of the network or violates the rules and\n"
"  protocols for communication across the network.\n"
"\n"
"    Corresponding Source conveyed, and Installation Information provided,\n"
"  in accord with this section must be in a format that is publicly\n"
"  documented (and with an implementation available to the public in\n"
"  source code form), and must require no special password or key for\n"
"  unpacking, reading or copying.\n"
"\n"
"    7. Additional Terms.\n"
"\n"
"    \"Additional permissions\" are terms that supplement the terms of this\n"
"  License by making exceptions from one or more of its conditions.\n"
"  Additional permissions that are applicable to the entire Program shall\n"
"  be treated as though they were included in this License, to the extent\n"
"  that they are valid under applicable law.  If additional permissions\n"
"  apply only to part of the Program, that part may be used separately\n"
"  under those permissions, but the entire Program remains governed by\n"
"  this License without regard to the additional permissions.\n"
"\n"
"    When you convey a copy of a covered work, you may at your option\n"
"  remove any additional permissions from that copy, or from any part of\n"
"  it.  (Additional permissions may be written to require their own\n"
"  removal in certain cases when you modify the work.)  You may place\n"
"  additional permissions on material, added by you to a covered work,\n"
"  for which you have or can give appropriate copyright permission.\n"
```

```
"\n"
"    Notwithstanding any other provision of this License, for material you\n"
"  add to a covered work, you may (if authorized by the copyright holders of\n"
"  that material) supplement the terms of this License with terms:\n"
"\n"
"      a) Disclaiming warranty or limiting liability differently from the\n"
"      terms of sections 15 and 16 of this License; or\n"
"\n"
"      b) Requiring preservation of specified reasonable legal notices or\n"
"      author attributions in that material or in the Appropriate Legal\n"
"      Notices displayed by works containing it; or\n"
"\n"
"      c) Prohibiting misrepresentation of the origin of that material, or\n"
"      requiring that modified versions of such material be marked in\n"
"      reasonable ways as different from the original version; or\n"
"\n"
"      d) Limiting the use for publicity purposes of names of licensors or\n"
"      authors of the material; or\n"
"\n"
"      e) Declining to grant rights under trademark law for use of some\n"
"      trade names, trademarks, or service marks; or\n"
"\n"
"      f) Requiring indemnification of licensors and authors of that\n"
"      material by anyone who conveys the material (or modified versions of\n"
"      it) with contractual assumptions of liability to the recipient, for\n"
"      any liability that these contractual assumptions directly impose on\n"
"      those licensors and authors.\n"
"\n"
"    All other non-permissive additional terms are considered \"further\n"
"  restrictions\" within the meaning of section 10.  If the Program as you\n"
"  received it, or any part of it, contains a notice stating that it is\n"
"  governed by this License along with a term that is a further\n"
"  restriction, you may remove that term.  If a license document contains\n"
"  a further restriction but permits relicensing or conveying under this\n"
"  License, you may add to a covered work material governed by the terms\n"
"  of that license document, provided that the further restriction does\n"
"  not survive such relicensing or conveying.\n"
"\n"
"    If you add terms to a covered work in accord with this section, you\n"
"  must place, in the relevant source files, a statement of the\n"
"  additional terms that apply to those files, or a notice indicating\n"
"  where to find the applicable terms.\n"
"\n"
"    Additional terms, permissive or non-permissive, may be stated in the\n"
"  form of a separately written license, or stated as exceptions;\n"
"  the above requirements apply either way.\n"
"\n"
"    8. Termination.\n"
"\n"
"    You may not propagate or modify a covered work except as expressly\n"
"  provided under this License.  Any attempt otherwise to propagate or\n"
"  modify it is void, and will automatically terminate your rights under\n"
"  this License (including any patent licenses granted under the third\n"
"  paragraph of section 11).\n"
"\n"
"    However, if you cease all violation of this License, then your\n"
"  license from a particular copyright holder is reinstated (a)\n"
"  provisionally, unless and until the copyright holder explicitly and\n"
"  finally terminates your license, and (b) permanently, if the copyright\n"
"  holder fails to notify you of the violation by some reasonable means\n"
"  prior to 60 days after the cessation.\n"
"\n"
"    Moreover, your license from a particular copyright holder is\n"
"  reinstated permanently if the copyright holder notifies you of the\n"
"  violation by some reasonable means, this is the first time you have\n"
"  received notice of violation of this License (for any work) from that\n"
"  copyright holder, and you cure the violation prior to 30 days after\n"
"  your receipt of the notice.\n"
"\n"
"    Termination of your rights under this section does not terminate the\n"
"  licenses of parties who have received copies or rights from you under\n"
"  this License.  If your rights have been terminated and not permanently\n"
"  reinstated, you do not qualify to receive new licenses for the same\n"
"  material under section 10.\n"
"\n"
"    9. Acceptance Not Required for Having Copies.\n"
"\n"
"    You are not required to accept this License in order to receive or\n"
"  run a copy of the Program.  Ancillary propagation of a covered work\n"
"  occurring solely as a consequence of using peer-to-peer transmission\n"
"  to receive a copy likewise does not require acceptance.  However,\n"
```

```
"  nothing other than this License grants you permission to propagate or\n"
"  modify any covered work.  These actions infringe copyright if you do\n"
"  not accept this License.  Therefore, by modifying or propagating a\n"
"  covered work, you indicate your acceptance of this License to do so.\n"
"\n"
"    10. Automatic Licensing of Downstream Recipients.\n"
"\n"
"    Each time you convey a covered work, the recipient automatically\n"
"  receives a license from the original licensors, to run, modify and\n"
"  propagate that work, subject to this License.  You are not responsible\n"
"  for enforcing compliance by third parties with this License.\n"
"\n"
"    An \"entity transaction\" is a transaction transferring control of an\n"
"  organization, or substantially all assets of one, or subdividing an\n"
"  organization, or merging organizations.  If propagation of a covered\n"
"  work results from an entity transaction, each party to that\n"
"  transaction who receives a copy of the work also receives whatever\n"
"  licenses to the work the party's predecessor in interest had or could\n"
"  give under the previous paragraph, plus a right to possession of the\n"
"  Corresponding Source of the work from the predecessor in interest, if\n"
"  the predecessor has it or can get it with reasonable efforts.\n"
"\n"
"    You may not impose any further restrictions on the exercise of the\n"
"  rights granted or affirmed under this License.  For example, you may\n"
"  not impose a license fee, royalty, or other charge for exercise of\n"
"  rights granted under this License, and you may not initiate litigation\n"
"  (including a cross-claim or counterclaim in a lawsuit) alleging that\n"
"  any patent claim is infringed by making, using, selling, offering for\n"
"  sale, or importing the Program or any portion of it.\n"
"\n"
"    11. Patents.\n"
"\n"
"    A \"contributor\" is a copyright holder who authorizes use under this\n"
"  License of the Program or a work on which the Program is based.  The\n"
"  work thus licensed is called the contributor's \"contributor version\".\n"
"\n"
"    A contributor's \"essential patent claims\" are all patent claims\n"
"  owned or controlled by the contributor, whether already acquired or\n"
"  hereafter acquired, that would be infringed by some manner, permitted\n"
"  by this License, of making, using, or selling its contributor version,\n"
"  but do not include claims that would be infringed only as a\n"
"  consequence of further modification of the contributor version.  For\n"
"  purposes of this definition, \"control\" includes the right to grant\n"
"  patent sublicenses in a manner consistent with the requirements of\n"
"  this License.\n"
"\n"
"    Each contributor grants you a non-exclusive, worldwide, royalty-free\n"
"  patent license under the contributor's essential patent claims, to\n"
"  make, use, sell, offer for sale, import and otherwise run, modify and\n"
"  propagate the contents of its contributor version.\n"
"\n"
"    In the following three paragraphs, a \"patent license\" is any express\n"
"  agreement or commitment, however denominated, not to enforce a patent\n"
"  (such as an express permission to practice a patent or covenant not to\n"
"  sue for patent infringement).  To \"grant\" such a patent license to a\n"
"  party means to make such an agreement or commitment not to enforce a\n"
"  patent against the party.\n"
"\n"
"    If you convey a covered work, knowingly relying on a patent license,\n"
"  and the Corresponding Source of the work is not available for anyone\n"
"  to copy, free of charge and under the terms of this License, through a\n"
"  publicly available network server or other readily accessible means,\n"
"  then you must either (1) cause the Corresponding Source to be so\n"
"  available, or (2) arrange to deprive yourself of the benefit of the\n"
"  patent license for this particular work, or (3) arrange, in a manner\n"
"  consistent with the requirements of this License, to extend the patent\n"
"  license to downstream recipients.  \"Knowingly relying\" means you have\n"
"  actual knowledge that, but for the patent license, your conveying the\n"
"  covered work in a country, or your recipient's use of the covered work\n"
"  in a country, would infringe one or more identifiable patents in that\n"
"  country that you have reason to believe are valid.\n"
"\n"
"    If, pursuant to or in connection with a single transaction or\n"
"  arrangement, you convey, or propagate by procuring conveyance of, a\n"
"  covered work, and grant a patent license to some of the parties\n"
"  receiving the covered work authorizing them to use, propagate, modify\n"
"  or convey a specific copy of the covered work, then the patent license\n"
"  you grant is automatically extended to all recipients of the covered\n"
"  work and works based on it.\n"
"\n"
"    A patent license is \"discriminatory\" if it does not include within\n"
```

```
"  the scope of its coverage, prohibits the exercise of, or is\n"
"  conditioned on the non-exercise of one or more of the rights that are\n"
"  specifically granted under this License.  You may not convey a covered\n"
"  work if you are a party to an arrangement with a third party that is\n"
"  in the business of distributing software, under which you make payment\n"
"  to the third party based on the extent of your activity of conveying\n"
"  the work, and under which the third party grants, to any of the\n"
"  parties who would receive the covered work from you, a discriminatory\n"
"  patent license (a) in connection with copies of the covered work\n"
"  conveyed by you (or copies made from those copies), or (b) primarily\n"
"  for and in connection with specific products or compilations that\n"
"  contain the covered work, unless you entered into that arrangement,\n"
"  or that patent license was granted, prior to 28 March 2007.\n"
"\n"
"    Nothing in this License shall be construed as excluding or limiting\n"
"  any implied license or other defenses to infringement that may\n"
"  otherwise be available to you under applicable patent law.\n"
"\n"
"    12. No Surrender of Others' Freedom.\n"
"\n"
"    If conditions are imposed on you (whether by court order, agreement or\n"
"  otherwise) that contradict the conditions of this License, they do not\n"
"  excuse you from the conditions of this License.  If you cannot convey a\n"
"  covered work so as to satisfy simultaneously your obligations under this\n"
"  License and any other pertinent obligations, then as a consequence you may\n"
"  not convey it at all.  For example, if you agree to terms that obligate you\n"
"  to collect a royalty for further conveying from those to whom you convey\n"
"  the Program, the only way you could satisfy both those terms and this\n"
"  License would be to refrain entirely from conveying the Program.\n"
"\n"
"    13. Use with the GNU Affero General Public License.\n"
"\n"
"    Notwithstanding any other provision of this License, you have\n"
"  permission to link or combine any covered work with a work licensed\n"
"  under version 3 of the GNU Affero General Public License into a single\n"
"  combined work, and to convey the resulting work.  The terms of this\n"
"  License will continue to apply to the part which is the covered work,\n"
"  but the special requirements of the GNU Affero General Public License,\n"
"  section 13, concerning interaction through a network will apply to the\n"
"  combination as such.\n"
"\n"
"    14. Revised Versions of this License.\n"
"\n"
"    The Free Software Foundation may publish revised and/or new versions of\n"
"  the GNU General Public License from time to time.  Such new versions will\n"
"  be similar in spirit to the present version, but may differ in detail to\n"
"  address new problems or concerns.\n"
"\n"
"    Each version is given a distinguishing version number.  If the\n"
"  Program specifies that a certain numbered version of the GNU General\n"
"  Public License \"or any later version\" applies to it, you have the\n"
"  option of following the terms and conditions either of that numbered\n"
"  version or of any later version published by the Free Software\n"
"  Foundation.  If the Program does not specify a version number of the\n"
"  GNU General Public License, you may choose any version ever published\n"
"  by the Free Software Foundation.\n"
"\n"
"    If the Program specifies that a proxy can decide which future\n"
"  versions of the GNU General Public License can be used, that proxy's\n"
"  public statement of acceptance of a version permanently authorizes you\n"
"  to choose that version for the Program.\n"
"\n"
"    Later license versions may give you additional or different\n"
"  permissions.  However, no additional obligations are imposed on any\n"
"  author or copyright holder as a result of your choosing to follow a\n"
"  later version.\n"
"\n"
"    15. Disclaimer of Warranty.\n"
"\n"
"    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY\n"
"  APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT\n"
"  HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM \"AS IS\" WITHOUT WARRANTY\n"
"  OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,\n"
"  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR\n"
"  PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM\n"
"  IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF\n"
"  ALL NECESSARY SERVICING, REPAIR OR CORRECTION.\n"
"\n"
"    16. Limitation of Liability.\n"
"\n"
"    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING\n"
```

```
"  WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS\n"
"  THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY\n"
"  GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE\n"
"  USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF\n"
"  DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD\n"
"  PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),\n"
"  EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF\n"
"  SUCH DAMAGES.\n"
"\n"
"    17. Interpretation of Sections 15 and 16.\n"
"\n"
"    If the disclaimer of warranty and limitation of liability provided\n"
"  above cannot be given local legal effect according to their terms,\n"
"  reviewing courts shall apply local law that most closely approximates\n"
"  an absolute waiver of all civil liability in connection with the\n"
"  Program, unless a warranty or assumption of liability accompanies a\n"
"  copy of the Program in return for a fee.\n"
"\n"
"                       END OF TERMS AND CONDITIONS\n"

//  #endif // _LICENSE_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : smart_ptr.h
//  file-create-date : Tue 28-Aug-2007 15:43 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : switch easily between TR1 and Boost smart pointers / header
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/smart_ptr.h $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This file allows one to switch easily between TR1 and Boost
//  smart pointers.

//  HEADER GUARD

#ifndef _SMART_PTR_H_
#define _SMART_PTR_H_

// ----------------------------------------------------------
//  setup           : smart pointer library
// ----------------------------------------------------------
//
//  Design notes
//
//      This file allows alternative shared pointer libraries
//      and/or headers to be switched in and out without
//      significant code changes.
//
//      The current setting (at the time of writing) uses TR1
//      shared pointers in the std::tr1:: namespace as included
//      by way of the <tr1/memory> header.
//
//      TR1 stands for Technical Report 1 and indicates libraries
//      that are very likely to be included in the next round of
//      C++ ISO standards process.
//
//      The shared pointer implementation can also be set to
//      Boost shared pointers (one reason might be that the
//      Boost.Serialization 1.34.1 (de)serialization library does
//      not support TR1 shared pointers).
//
//      The <tr1/memory> header may well migrate to <memory> in
//      the not too distant future.  If so, one can speculate
//      that a move to the std:: namespace would also occur.
//
//  Header usage
//
//      Use this file as follows (subdir paths may also be needed):
//
//      #include "smart_ptr.h"  // toggle between TR1 and Boost smart pointers
//      #include "common.h"     // common definitions for project (place last)
//
```

```
//  Compile-time overwriting of the 'XE_SHARED_PTR' macro
//
//      The 'XE_SHARED_PTR' macro can be explicitly set and
//      passed to the preprocessor at compile time -- otherwise
//      the default defined in this file will be utilized.
//
//  CAUTION: namespace qualification should not be used
//
//      Do not namespace qualify shared_ptr in the codebase --
//      UNLESS the flexibility given here is expressly not
//      wanted.  Hence the preferred syntax is:
//
//          shared_ptr<T> t;
//
//  CAUTION: 'make_shared' not supported by TR1
//
//      The free function template 'boost::make_shared<>' is not
//      supported by g++ 4.1.2 -- and therefore not used here.
//      (I did look at implementing it for TR1 but that would be
//      far from straightforward.)
//
//  CAUTION: TR1 smart pointers and Boost.Serialization cannot be mixed
//
//      See r920 of this file for further information.
//
//      This is no longer an issue because the
//      Boost.Serialization library is no longer employed.
//
//  References
//
//      Consult the GNU GCC compiler documentation for details of
//      the move from <tr1/memory> to <memory>.
//
//      The 'Boost - Users' email list is a good source of
//      information regarding the current status of Boost
//      libraries.
//
// ------------------------------------------------------------

//  set the smart pointer libraries to be used
//
//  0 = TR1 std::tr1::shared_ptr and similar (conflicts with Boost.Serialization)
//  1 = Boost.Smart_ptr boost::shared_ptr and similar

#if !defined(XE_SHARED_PTR)
# define XE_SHARED_PTR 0                    // set default
#endif

// ---------------------------------
//  TR1 shared pointer
// ---------------------------------

// see '/usr/include/c++/4.1.2/tr1' etc

#if   (XE_SHARED_PTR == 0)

#include <tr1/memory>                       // TR1 shared pointer

// CAUTION: [1] see r3179 for the unified diffs, essentially
//
//   '/usr/include/c++/4.1.2/tr1/memory'             : duly copied, revised link added
//   '/usr/include/c++/4.1.2/tr1/boost_shared_ptr.h' : duly copied, 'virtual' in one place
//
// Follow up: this hack is no longer required somewhere between
// GCC 4.1.2 and 4.4.3.  That said, the '_XWEFF' code has been
// left in place but it could be ripped out sometime.

  using std::tr1::shared_ptr;
  using std::tr1::weak_ptr;
  using std::tr1::enable_shared_from_this;
  using std::tr1::const_pointer_cast;
  using std::tr1::static_pointer_cast;
  using std::tr1::dynamic_pointer_cast;
  using std::tr1::bad_weak_ptr;                   // exception class

// ---------------------------------
//  Boost.Smart_ptr shared pointer
// ---------------------------------

// see '/usr/local/include/boost-1_39' etc
```

```
#elif (XE_SHARED_PTR == 1)

# include <boost/shared_ptr.hpp>                  // Boost shared pointer (serializable)
# include <boost/weak_ptr.hpp>                    // Boost weak pointer (serializable)
# include <boost/enable_shared_from_this.hpp>     // Boost smart pointer support
// # include <boost/make_shared.hpp>              // Boost make_shared factory function

  using boost::shared_ptr;
  using boost::weak_ptr;
  using boost::enable_shared_from_this;
  using boost::const_pointer_cast;
  using boost::static_pointer_cast;
  using boost::dynamic_pointer_cast;
  using boost::bad_weak_ptr;                      // exception class
  // using boost::make_shared;                    // factory function

#else
# error "XE_SHARED_PTR macro not set to a supported value {0,1}"
#endif

#endif // _SMART_PTR_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : ospmodes.h
//  file-create-date : Thu 05-Feb-2009 23:08 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain mode enums / header (only)
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/ospmodes.h $

//  HEADER GUARD

#ifndef _OSPMODES_H_
#define _OSPMODES_H_

//  LOCAL AND SYSTEM INCLUDES

//  none required

//  CODE

// ------------------------------------------------------------
//  ENUM            : xeona::DomainMode
// ------------------------------------------------------------
//
//  Design notes
//
//      This enum embeds two pieces of information: the capacity
//      mode and the commitment mode.
//
//      This allows a single argument to be passed around,
//      thereby simplifying the coding (although two arguments or
//      a std::pair could equally well have been used).
//
//      Capacity modes
//
//          Capacity modes deal with the way capacities are set
//          on gateways.  Withholding need not be commercially
//          strategic, but is often so.
//
//      Commitment modes
//
//          Commitment modes deal with the way transactions are
//          solved within control domains.
//
//  CAUTION: do no hardcode integer values
//
//      NEVER EVER use the underlying integer value in client
//      code -- ALWAYS use the respective enumerator (that is,
//      the enumeration identifier) instead!  It is expected that
//      the values assigned here can be changed at will!
//
//  CAUTION: 15 enumerators only
//
```

```
//      To play safe, the maximum enumerator value should not
//      exceed 16384 -- meaning a limit of 15 values, counting
//      any gaps but excluding 'e_modeNotSpecified'.
//
//      Lischner (2003 pp26-27) discusses the integer ranges for
//      enums in considerable detail (the key message is that the
//      compiler chooses the underlying type from 'signed char'
//      to 'unsigned int').
//
//  Committment modes in detail
//
//      The commitment mode determines in part how the
//      optimization sub-problems will be uploaded and
//      interpreted.
//
//      Values for this enum are held within the various
//      'TechnicalAsset' and 'AssetOperator' sub-classes.
//
//      This value is, in turn, sent thru to the derived
//      'OptimSubProb' instances so that they can respond
//      appropriately.
//
//      In addition, integrity checks are carried out to ensure
//      the selected commitment strategy is consistent with the
//      various domain-specific assets and objects present (this
//      being a modeling issue and not a coding error).
//
//      Note also that it will probably be rather difficult to
//      add to the current cost types (fin, ghg, nox, dep, luc)
//      and commitment modes (shortrunFin, etc).  Both concepts
//      are deeply embedded within the design of 'xeona'.
//
// ----------------------------------------------------------

namespace xeona
{
  enum DomainMode                   // CAUTION: add new enums to the or'ed lists below
    {
      e_modeNotSpecified =     0,

      // capacity modes

      e_usePresets       =     1,  // use XEM-input capacities, no run-time calculation *
      e_withholdOkay     =     2,  // withholding acceptable
      e_withholdBan      =     4,  // withholding banned, run at normal technical capacity
      e_crisisOperation  =     8,  // relax all further non-mandatory constraints *

      // CAUTION: * = not supported by the codebase as it stands,
      // nonetheless provided for possible future use

      // commitment modes

      e_shortrunFin      =    32,  // short-run financial cost minimization
      e_shortrunGhg      =    64,  // short-run GHG contribution minimization
      e_shortrunNox      =   128,  // short-run NOx contribution minimization
      e_shortrunDep      =   256,  // short-run depletable resource use minimization
      e_shortrunLuc      =   512,  // short-run land use minimization

      e_auctionLmp       =  2048,  // locational marginal (nodal) pricing auction
      e_adminMerit       =  4096,  // prescribed merit order
      e_adminFirst       =  8192,  // first feasible solution

      // define some mode sums -- especially useful for 'xeona::isTwoContained' calls

      e_normalCapModes   =  e_withholdOkay     |
                            e_withholdBan,

      e_crisisCapModes   =  e_crisisOperation,

      e_capacityModes    =  e_usePresets       |
                            e_normalCapModes   |
                            e_crisisCapModes,

      e_shortrunModes    =  e_shortrunFin      |
                            e_shortrunGhg      |
                            e_shortrunNox      |
                            e_shortrunDep      |
                            e_shortrunLuc,

      e_commitmentModes  =  e_shortrunModes    |
                            e_auctionLmp       |
```

```
                              e_adminMerit       |
                              e_adminFirst,

      e_all              =    e_capacityModes    |
                              e_commitmentModes,

      e_maxAggregate     =    32768            // 2**15
    };

} // namespace 'xeona'

#endif // _OSPMODES_H_

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : common.cc
//  file-create-date : Sun 08-Apr-2007 05:43 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : provide useful definitions to the entire codebase / implementation
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/common.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This file defines common items that may be needed throughout
//  the application.
//
//  Assuming the normal makefile is used, this file is always
//  "touched" in order to ensure it is remade and therefore
//  fully current.
//
//  The compiler macros '__linux__' and '__unix__' are used
//  exclusively in this codebase.  A more robust approach would
//  be to utilize instead:
//
//  // #ifdef __linux__
// #if defined(linux) || defined(__linux) || defined(__linux__)

//  LOCAL AND SYSTEM INCLUDES

#include "common.h"           // companion header for this file (place first)

#include "a/logger.h"         // run-time logging functionality (as required)

#include <string>             // C++ strings

// specific sources of information

#include <boost/math/constants/constants.hpp>     // high-precision mathematical constants
#include <boost/version.hpp>                      // Boost release information
#include <cstdlib>                                // EXIT_SUCCESS macro
#include <glpk.h>                                 // GNU GLPK version information

//  NAMESPACE : xeona

namespace xeona
{
  //  RUN-TIME FLAGS

#ifdef _XUTEST
  bool nopro = true;                  // unit tests require 'true'
#else
  bool nopro = false;                 // can be reset via '--krazy' to 'true'
#endif

  bool again      = false;            // set via '--again' to continue after bad solve
```

```
  bool pepper     = false;              // set via '--pepper' for random reseeding
  bool zero       = true;               // set via '--zero' for disabling close-to-zero
  unsigned yeek   = 0;                  // set via '--yeek' for selecting test code

  //  DEFINITIONS: debug flag
  //  controlled by compile-time _XDEBUG macro

#ifdef _XDEBUG                                  // xeona macro
  const bool DBUG = true;
#else
  const bool DBUG = false;
#endif

  //  controlled by compile-time NDEBUG macro

#ifdef NDEBUG                                   // official macro
  const bool NBUG = true;
#else
  const bool NBUG = false;
#endif

  //  DEFINITIONS: process id

  pid_t pid = 0;                                // underlying 'unsigned', refer <unistd.h>

  //  DEFINITIONS : application exit statuses
  //  consider existing user expectations when modifying

  const int exit_success              =   0; // for unit testers only
  const int exit_fail                 =   1; // for unit testers only

  const int ret_not_overwritten       = 255; // highest possible return integer

  const int ret_success               =   0; // general success
  const int ret_message               =   0; // display and quit returns
  const int ret_noclass               =   8; // class not found
  const int ret_usage                 =   9; // command-line usage error
  const int ret_fail                  = 255; // serious failure

  const int exit_kill_on_log          =  10; // see 'a/exitstat.cc' for descriptions
  const int exit_empty_wrap           =  11;
  const int exit_non_registration     =  12;
  const int exit_short_timeseries     =  13;
  const int exit_file_not_found       =  14;
  const int exit_xem_data_issue       =  15;
  const int exit_lazy_link_fail       =  16;
  const int exit_cannot_run_guard     =  17;
  const int exit_bad_authorship       =  18;
  const int exit_full_link_fail       =  19;
  const int exit_empty_field_on_write =  20;
  const int exit_timeseries_not_found =  21;
  const int exit_yeek_abandon         =  22;

  const int exit_entity_fail          =  30;

  const int exit_bad_assign_cast      =  40;
  const int exit_boost_exception      =  41;
  const int exit_std_out_of_range     =  42;
  const int exit_std_logic_error      =  43;
  const int exit_std_bad_alloc        =  44;
  const int exit_std_exception        =  45;
  const int exit_unknown_exception    =  46;

  const int ret_model_file_fault      =  50;
  const int ret_infeasibility         =  51;
  const int ret_errant_simulation     =  52;
  const int ret_test_code_used        =  53;
  const int ret_other                 =  58;
  const int ret_status_not_known      =  60;

  const int exit_test99               =  99;
  const int exit_test100              = 100; // special 100 return for test code

  const int exitTripLevelDefault   = logga::kill;      // success means no KILL logging

  //  DEFINITIONS: hand-set build information
  //  update as appropriate

  const std::string buildCopyright = "2007 - 2010 Robbie Morrison";
  const std::string buildTitle     = "xeona energy systems modeling environment";
  const std::string codebaseStatus = "work-in-progress"; // later use "alpha testing"
```

```
  //  DEFINITIONS : auto-set build information
  //  controlled by compile-time macros

#ifdef _XSVNREV
  const int svnRev = _XSVNREV;
#else
  const int svnRev = 0;
#endif

  const int gccVerMajor     = __GNUC__;
  const int gccVerMinor     = __GNUC_MINOR__;
  const int gccPatchLevel   = __GNUC_PATCHLEVEL__;

  const int glpkVerMajor    = GLP_MAJOR_VERSION;
  const int glpkVerMinor    = GLP_MINOR_VERSION;

  const int boostVersionNum         = BOOST_VERSION;        // 103401
  const std::string boostVersionStr = BOOST_LIB_VERSION;    // "1_34_1"

#if   defined (__linux__)
  const std::string osName  = "Linux";       // Linux
#elif defined (__unix__)
  const std::string osName  = "UNIX";        // generic UNIX
# else
  const std::string osName  = "(not Linux or UNIX)";
#endif

#ifdef _XRELEASE                             // xeona macro
  const bool releaseStatus = true;
#else
  const bool releaseStatus = false;
#endif

  //  DEFINITIONS: compiler macros
  //  compiler macro __DATE__ yields Mar  8 2007
  //  compiler macro __TIME__ yields 23:00:59

  // CAUTION: omit double quotes on __DATE__ and __TIME__
  const std::string buildDate = __DATE__;
  const std::string buildTime = __TIME__;

  //  DEFINITIONS : system-specific literals

  // CAUTION: this code has NOT BEEN TESTED on either 32-bit or 64-bit Windows
#if defined (WIN32) || defined (_WIN32) || defined (__WIN32__) || defined (_WIN64)
  const char osSlash = '\\';                 // CAUTION: escaping is needed
#else
  const char osSlash = '/';
#endif

  //  DEFINITIONS: miscellaneous constants

  //  GENERAL : define the units used in reporting -- these are
  //  SHALLOW settings which affect the printed output but which
  //  have NO INFLUENCE on the underlying calculations

  const std::string iso4217 = "EUR";    // ISO 4217 three char code (for example, EUR USD)
  const std::string logfileDefault = "hardcoded.log";  // hardcoded logfile name

  //  DEFINITIONS: mathematical and physical constants

  // pi = 3.1415926535897932384...
  // e  = 2.71828182845904523536...

  const double mathsPi          = boost::math::constants::pi<double>();
  const double mathsE           = boost::math::constants::e<double>();

  //  source: Wikipedia, July 2008

  const double stdAtmosphere    = +101325;        // [Pa]
  const double stdGravity       = +9.80665;       // [m/s2]
  const double absoluteZero     = -273.15;        // [degrees C]

  const int secondsPerYear      = 8760 * 3600;    // seconds in 365 day year

  //  DEFINITIONS: model files

  const std::string backupTag           = "~";         // append tag for file backup
  const std::string modelExt            = ".xem";      // xeona model extension
  const std::string modelStubDefault    = "test";      // default model name
```

```
  const std::string modelInbuiltDefault = "inbuilt";   // default inbuilt model name
  const std::string modelGuardTag       = ".guard";    // tag indicating guard file

  const std::string modelDisableChar = "#";  // record or field disable string
  const std::string modelTsRepeater  = ".."; // timeseries repeat indicator
  const std::string modelStringDelim = "\""; // pairwise delimiter for string values [1]
  const std::string modelBidDelim    = "*";  // nodal bid separator
  const std::string modelEndMarker   = "model-end";    // model end marker

  const unsigned    modelFieldIndent =  4;   // field line indent
  const unsigned    modelAngleIndent = 45;   // miniumum field angle char allowance

  // [1] successfully tested with "\"" and "'" whilst noting that "" is illegal

  //  DEFINITIONS: optimization problem (OSP) labeling

  const std::string ospTagSep = ".";         // separator string used by class 'Label' [1]

  // [1] a dot '.' will conflict with float representations while a "::" uses more room

  //  DEFINITIONS: mandatory entity identifiers

  const std::string timehorizon = "time-horizon";      // 'TimeHorizon' entity
  const std::string overseer    = "overseer";          // 'Overseer' entity

  //  DEFINITIONS: run-time behavior

#ifdef _XDEBUG                               // xeona macro
  const unsigned reportLevelOpening  = 4;    // opening reporting before cmd-line parsing
  const unsigned reportLevelDefault  = 6;    // default reporting 0-6 (quiet - verbose)
  const unsigned beepModeDefault     = 0;    // default beep behavior 0-2 (silent - loud)
#else
  const unsigned reportLevelOpening  = 2;
  const unsigned reportLevelDefault  = 2;
  const unsigned beepModeDefault     = 2;
#endif
  const unsigned sameLogLimit        = 3;    // limit repetitive log messages
  const unsigned consoleWidth   = _XTCOLS;   // macro set via CPPFLAGS [1]

  // [1] 'sojus.fb10.tu-berlin.de', with a scroll bar, supports 143 chars
  // to check $ stty --all # take columns value OR $ stty size # take second value
  // to test: $ for i in $(seq 150); do printf "%${i}d\n" $i; done

  //  DEFINITIONS: external programs

  const std::string webbrowser = "firefox";  // invocation string with options [1]

  // [1] used by 'GlpkViz' class to display GLPK problem instances

  //  DEFINITIONS: xedocs text file

#ifndef _XUTEST
# define XE_XEDOCS_FILE "a/xedocs.txt"
#else
# define XE_XEDOCS_FILE "a/xedocs.ut.txt"
#endif

  const std::string xedocsFileName = XE_XEDOCS_FILE;
  const std::string xedocsFileContents =
#include XE_XEDOCS_FILE                     // CAUTION: file must exist at compile time
  ;                                          // CAUTION: final semicolon is necessary

#undef XE_XEDOCS_FILE                        // undefine macro for defensive programming

} // namespace xeona

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : appinfo.cc
//  file-create-date : Mon 16-Apr-2007 12:07 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : start and end-of-run reporting / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/appinfo.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "appinfo.h"          // companion header for this file (place first)

#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../a/yeek.h"        // yeek (for running extra code) value interpretation
#include "../a/exitstat.h"    // exit status database

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <locale>             // locale specific information
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <cstdlib>            // C-style exit(), getenv(), system(), NULL, EXIT_SUCCESS
#include <unistd.h>           // POSIX sleep(), usleep(), access(), chown()

#include <boost/algorithm/string.hpp>        // string recasing, trimming, splitting
#include <boost/format.hpp>                  // printf style formatting
#include <boost/lexical_cast.hpp>            // lexical_cast<>()
#include <boost/tokenizer.hpp>               // string tokenizer
#include <boost/foreach.hpp>                 // BOOST_FOREACH iteration macro

// CAUTION: 'boost/date_time/posix_time/posix_time' DOES require
// the -lboost_date_time linking directive in the makefile
// (although some uses of this header may not)

#include <boost/date_time/posix_time/posix_time.hpp>   // plus io

//  CODE

// ------------------------------------------------------------
//  STRING CONSTANT : ::licenseText
// ------------------------------------------------------------
//  See Becker (2007 p350) for a similar example.
// ------------------------------------------------------------

namespace
{
  const std::string licenseText =
#include "license.h"              // contains double-quoted newline-terminated text
  ;                               // final semicolon
```

```
}

// ----------------------------------------------------------
//  CLASS           : AppInfo
// ----------------------------------------------------------
//  Purpose      : supply build and run information
//  Status       : complete
//
//  Design notes
//
//      Volatile information is held in 'common.cc'.  And
//      'common.cc' is either touched or modified (for a release
//      build) during make and is thus always current.
//
//      Some formatting routines could be better refactored using
//      Boost library calls.
//
//      Utility function beginning 'show' generally contain
//      newlines, whereas utility functions beginning 'get'
//      generally do not.
//
//      The terminology relating to files derives from
//      Boost.Filesystem.  Read that documentation for more
//      information.
//
// ----------------------------------------------------------

// CREATORS

AppInfo::AppInfo
(int                            argc,         // from main()
 char**                         argv,         // from main()
 const boost::filesystem::path initial) :    // from main()
   d_tic(boost::posix_time::microsec_clock::universal_time()),
   d_progPath(pathFromString(argv[0])),      // utility function
   d_startPath(initial),
   d_fullCmdLine(argProcess(argc, argv)),    // utility function
   d_clReportLevel(0),                       // set later by Boost.Program_options
   d_clModelName()                           // set later by Boost.Program_options
{
  // initial reporting
  s_logger->repx(logga::dbug, "constructor call, argv[0]", argv[0]);
  s_logger->repx(logga::dbug, "d_progPath as leaf", extractLeaf(d_progPath));

  // recover process id
  xeona::pid = getpid();                     // refer <unistd.h>
  s_logger->repx(logga::dbug, "getpid call", xeona::pid);

  // path (including leaf) reporting
  std::ostringstream put;
  put << "  full path (stream) : " << d_progPath              << "\n"
      << "  full path (native) : " << extractPath(d_progPath) << "\n";
  s_logger->putx(logga::dbug, put);
}

AppInfo::~AppInfo()
{
  s_logger->repx(logga::dbug, "destructor call", "");
}

// ACCESSORS

const std::string
AppInfo::getProgramName() const
{
  return extractLeaf(d_progPath);
}

const std::string
AppInfo::getBuildDate() const
{
  // CAUTION: this function was originally written using
  // 'boost::posix_time' and 'boost::gregorian' objects -- but
  // there were memory problems, as detected by valgrind, when
  // using the relevant streaming operators.

  std::string date = xeona::buildDate;  // from compiler macro __DATE__ set in common.cc

  // tokenize and create new date string

  // NOTE: the 'boost::tokenizer' class has a fairly limited
```

```
  // interface and, in particular, no subscript or [] operator --
  // hence the need to create an iterator to gain access to the
  // newly found tokens

  boost::tokenizer<> tok(date);                              // defaults used
  boost::tokenizer<>::iterator grab = tok.begin();
  std::string smonth =   *grab;                              // Mar
  std::string sday   = *++grab;                              // 8
  std::string syear  = *++grab;                              // 2007

  std::string zmonth;
  if ( smonth == "Jan" ) zmonth = "01";
  if ( smonth == "Feb" ) zmonth = "02";
  if ( smonth == "Mar" ) zmonth = "03";
  if ( smonth == "Apr" ) zmonth = "04";
  if ( smonth == "May" ) zmonth = "05";
  if ( smonth == "Jun" ) zmonth = "06";
  if ( smonth == "Jul" ) zmonth = "07";
  if ( smonth == "Aug" ) zmonth = "08";
  if ( smonth == "Sep" ) zmonth = "09";
  if ( smonth == "Oct" ) zmonth = "10";
  if ( smonth == "Nov" ) zmonth = "11";
  if ( smonth == "Dec" ) zmonth = "12";

  std::string zday   = zeroPadMe(sday, 2);
  std::string output = syear + "-" + zmonth + "-" + zday;
  return output;
}

const std::string
AppInfo::getBuildTime() const
{
  return xeona::buildTime;         // from compiler macro __TIME__ set in common.cc
}

const std::string
AppInfo::getCodebaseStatus() const
{
  return xeona::codebaseStatus;              // defined in common.cc
}

const std::string
AppInfo::getTitle() const
{
  return xeona::buildTitle;                  // defined in common.cc
}

const std::string
AppInfo::getCopyright() const
{
  return xeona::buildCopyright;              // defined in common.cc
}

const unsigned
AppInfo::getSvnRevision() const
{
  return xeona::svnRev;                      // from 'common.cc'
}

const std::string
AppInfo::getGccVersion() const
{
  const std::string gcc = boost::str(boost::format("%d.%d.%d")
                                     % xeona::gccVerMajor     // say  4
                                     % xeona::gccVerMinor     // say  1
                                     % xeona::gccPatchLevel); // say  2
  return gcc;
}

const std::string
AppInfo::getGlpkVersion() const
{
  // CAUTION: the following string should really be set
  // intelligently, probably via a preprocessor macro computed in
  // the makefile

#if 1 // 0 = runtime, 1 = static
  const std::string link = "static";         // statically linked
#else
  const std::string link = "runtime";        // dynamically linked
#endif // 0
```

```
  const std::string glpk = boost::str(boost::format("%d.%d %s")
                                      % xeona::glpkVerMajor    // say  4
                                      % xeona::glpkVerMinor    // say 19
                                      % link);
  return glpk;
}

const std::string
AppInfo::getBoostVersion() const
{
  // documentation from <boost/version.hpp> header
  //
  //  BOOST_VERSION % 100 is the sub-minor version
  //  BOOST_VERSION / 100 % 1000 is the minor version
  //  BOOST_VERSION / 100000 is the major version
  //
  // thus 103401 -> "1.34.1"

  const int boost_version = xeona::boostVersionNum;
  const int major    = boost_version / 100000;
  const int minor    = boost_version / 100 % 1000;
  const int subminor = boost_version % 100;

  // determine the linking strategy in relation to the
  // preprocessor macro '_XRELEASE'

  // CAUTION: the logic here must align with that used in the
  // 'mach' build script and/or the makefile

  const std::string linking = xeona::releaseStatus ? "static" : "dynamic";

  // build
  std::ostringstream oss;
  oss << major << "." << minor << "." << subminor << " " << linking;

  return oss.str();
}

const std::string
AppInfo::getOsInfo() const
{
  const std::string os = xeona::osName;
  return os;
}

// for information on the class 'std::locale', see Lischner (2003
// pp578-581) and, more generally, Josuttis (1999 pp692-726)

const std::string
AppInfo::getLocale() const
{
  // grab native locale
  std::locale local = std::locale();         // native locale, refer <locale>
                                             // as used by Boost.String_algo by default
  // check against "C" locale
  if ( local == std::locale::classic() )     // 'classic' is the "C" locale
    {
      return local.name();                   // type 'basic_string<char>'
    }
  else
    {
      return local.name() + " (\"C\" locale recommend)";
    }
}

const bool                                   // 'true' when built using 'mach -r' script
AppInfo::getReleaseStatus() const
{
  return xeona::releaseStatus;               // set via _XRELEASE preprocessor macro
}

const std::string
AppInfo::showSplash() const
{
  std::ostringstream ss;
  ss   << ""                                                    << "\n"
       << "  xeona"                                            << "\n"
       << ""                                                   << "\n"
       << "     " << getTitle()                                << "\n"
       << "     " << "copyright (c) " << getCopyright()         << "\n";
```

```
  if ( getReleaseStatus() )
    ss << "     " << "show GPL v3 license with option --legal"  << "\n";
  ss   << "     " << "all timestamps in UTC time"               << "\n";

  s_logger->addSmartBlank();                // next logger call should add a blank line
  s_logger->addFinalStdoutBlank();          // add a final blank to stdout

  return ss.str();
}

const std::string
AppInfo::showInfo
(const xeona::SimKind simKind,
 const unsigned       beepMode,
 const unsigned       exitTrip) const
{
  std::string buildStatus               = "non-release";
  if ( getReleaseStatus() ) buildStatus = "release";

  std::string debugInfo                 = "unset";
  if ( xeona::DBUG ) debugInfo          = "set";

  std::string ndebugInfo                = "unset";
  if ( xeona::NBUG ) ndebugInfo         = "set";

  std::string modelName;
  if ( d_clModelName.empty() )
    modelName = xeona::modelStubDefault + xeona::modelExt + " (defaulting)";
  else
    modelName = d_clModelName + "";

  std::string svnRevision;
  const unsigned rev = getSvnRevision();
  if ( rev == 0 ) svnRevision = "(not-in-sync)";
  else            svnRevision = boost::lexical_cast<std::string>(rev);

  std::string simulationMode = "(not overwritten)";
  std::ostringstream ossSimMode;
  switch ( simKind )
    {
    case xeona::e_notSpecified:
      ossSimMode << "(not specified)";
      break;
    case xeona::e_hollowCall:
      ossSimMode << "return after processing command-line";
      break;
    case xeona::e_identifyFile:
      ossSimMode << "identify file and quit";
      break;
    case xeona::e_parseModel:
      ossSimMode << "parse and shallow overwrite model";
      break;
    case xeona::e_invokeFactory:
      ossSimMode << "construct entities and deep overwrite model";
      break;
    case xeona::e_linkAndConnect:
      ossSimMode << "link and connect entities";
      break;
    case xeona::e_firstStepRun:
      ossSimMode << "first-step run";
      break;
    case xeona::e_fullRun:
      ossSimMode << "full run";
      break;
    default: std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
    }
  ossSimMode << " (" << simKind << ")";
  simulationMode = ossSimMode.str();

  std::string noproStatus         = "exit cleanly on fault";
  if ( xeona::nopro ) noproStatus = "run to completion or failure (and core dump)";

  std::string beeping = "(not overwritten)";
  std::ostringstream ossBeeping;
  switch ( beepMode )
    {
    case 0: ossBeeping << "silent";               break;
    case 1: ossBeeping << "on completion";        break;
    case 2: ossBeeping << "on displayed warnings"; break;
    default: std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
    }
```

```
  ossBeeping << " (" << beepMode << ")";
  beeping = ossBeeping.str();

  std::string triggerSource = s_logger->getTrigger();
  if ( triggerSource.empty() ) triggerSource = "(none)";

  std::string exitTripLevel = "(not overwritten)";
  std::ostringstream ossExitTripLevel;
  switch ( exitTrip )
    {
    case logga::yeek: ossExitTripLevel << "report trips not enabled";    break;
    case logga::kill: ossExitTripLevel << "on kill calls";              break;
    case logga::warn: ossExitTripLevel << "on warning calls upward";    break;
    case logga::info: ossExitTripLevel << "on information calls upward"; break;
    case logga::dbug: ossExitTripLevel << "on debug calls upward";      break;
    case logga::xtra: ossExitTripLevel << "on extra calls upward";      break;
    case logga::adhc: ossExitTripLevel << "on any report call";         break;
    default: std::clog << "** coding error 03 in source file " << __FILE__ << std::endl;
    }
  ossExitTripLevel << " (0 thru " << exitTrip << ")";
  exitTripLevel = ossExitTripLevel.str();

  std::string emptyData = "(not overwritten)";
  switch ( logga::rankNoData )
    {
    case logga::warn: emptyData = "warn if empty fields encountered"; break;
    case logga::dbug: emptyData = "(no special action)";              break;
    default: std::clog << "** coding error 04 in source file " << __FILE__ << std::endl;
    }

  std::ostringstream ss;
  ss << ""                                                 << "\n"
     << "  build information"                              << "\n"
     << ""                                                 << "\n"
     << "     compile date      :  " << getBuildDate()      << "\n"
     << "     operating system  :  " << getOsInfo()         << "\n"
     << "     g++ compiler      :  " << getGccVersion()     << "\n"
     << "     glpk solver       :  " << getGlpkVersion()    << "\n"
     << "     boost c++ libs    :  " << getBoostVersion()   << "\n"
     << "     build context     :  " << buildStatus         << "\n"
     << "     codebase status   :  " << getCodebaseStatus() << "\n"
     << "     terminal width    :  " << xeona::consoleWidth << "\n"
     << "     _XDEBUG app       :  " << debugInfo           << "\n"
     << "     NDEBUG c++ libs   :  " << ndebugInfo          << "\n"
     << "     svn revision      :  " << svnRevision         << "\n"
     << ""                                                 << "\n"
     << "  run information"                                 << "\n"
     << ""                                                 << "\n"
     << "     binary name       :  " << getProgramName()    << "\n"
     << "     logname           :  " << getLogname()        << "\n"
     << "     locale            :  " << getLocale()         << "\n"
     << "     start date        :  " << getCurrentDate()    << "\n"
     << "     start time (UTC)  :  " << getCurrentTime()    << "\n"
     << "     process ID        :  " << xeona::pid          << "\n"
     << ""                                                 << "\n"
     << "  command-line settings"                           << "\n"
     << ""                                                 << "\n"
     << "     command-line      :  " << d_fullCmdLine       << "\n"
     << "     defensive coding  :  " << noproStatus         << "\n"
     << "     beep behavior     :  " << beeping             << "\n"
     << "     reporting level   :  " << getReportLevel()    << "\n"
     << "     no data           :  " << emptyData           << "\n"
     << "     watch regex       :  " << triggerSource       << "\n"
     << "     nonzero exit trip :  " << exitTripLevel       << "\n"
     << "     simulation mode   :  " << simulationMode      << "\n"
     << "     given model name  :  " << modelName           << "\n";

  s_logger->addSmartBlank();                // next logger call should add blank line
  s_logger->addFinalStdoutBlank();          // add a final blank to stdout

  return ss.str();
}

const std::string
AppInfo::showHelp
(std::string strDesc) const
{
  // create string summary of exit statuses
  ExitStatus exitstatus;                    // accept default not-present message
  std::ostringstream buffer;
  for ( int i = 0; i <= 100; ++i )
```

```
    {
      std::string interpretation;
      if ( exitstatus(i, interpretation) )   // 'true' if present
        {
          buffer << boost::format("   %3d = %s") % i % interpretation << "\n";
        }
    }
  const std::string exitstatii = buffer.str();

  // principal string-stream
  std::ostringstream ss;
  ss
<< "\n"

<<"usage: "
<< getProgramName()
<< "  [options]  [model[.xem]]"
<< "\n"

<< "\n"

<< strDesc                              // generated by 'Boost.Program_options'

<< "\n"

<< "numeric arguments (defaults given above):\n"
<< "  beep\n"
<< "      0 = silent\n"
<< "      1 = on completion\n"
<< "      2 = plus on displayed warnings\n"
<< "  report\n"
<< "      0 = no output\n"
<< "      1 = show kills\n"
<< "      2 = plus warnings\n"
<< "      3 = plus information\n"
<< "      4 = plus debug\n"
<< "      5 = plus extra\n"
<< "      6 = plus adhoc\n"
<< "      7 = as 6 but do not truncate logging to fit terminal (ignore _XCOLS macro)\n"
<< "  mode\n"
<< "      1 = process command-line\n"
<< "      2 = plus identify model file\n"
<< "      3 = plus parse file with shallow (unprocessed string) overwrite\n"
<< "      4 = plus construct entities with deep (lexically cast and recast) overwrite\n"
<< "      5 = plus link (informational) and connect (physical) entities as required\n"
<< "      6 = plus undertake the first simulation step only\n"
<< "      7 = plus undertake the remaining simulation steps\n"

<< "\n"

<< "default model names:\n"
<< "      normal  = "
<< xeona::modelStubDefault << xeona::modelExt
<< "\n"
<< "      inbuilt = "
<< xeona::modelInbuiltDefault << xeona::modelExt
<< "\n"
<< "      guard   = "
<< xeona::modelStubDefault << xeona::modelGuardTag << xeona::modelExt
<< "\n"

<< "\n"

<< "options generally:\n"
<< "    options are processed in order listed above\n"
<< "    console reporting increases user/kernel context switching and slows execution\n"

<< "\n"

<< "options specifically:\n"
<< "    option --class + displays all documented entity class names"
<< " (+ is special character)\n"
<< "    option --class . displays all documented entity class data layouts (. is regex)\n"
<< "    option --inbuilt requires a step argument of 2 or more\n"
<< "    option --exittrip increases the sensitivity of failure based on reporting level\n"
<< "    option --inbuilt and option --guard cannot be used together\n"
<< "    option --krazy is provided for testing and can yield useful insights\n"
<< "    option --yeek is documented below\n"

<< "\n"
```

```
<< "output streams:\n"
<< "    stdout (file descriptor 1) = splash screens, help and usage messages"
<< ", caught exception notifications\n"
<< "    stderr (file descriptor 2) = logging output, caught exception explanations\n"

<< "\n"

<< "compiled-in literals which affect application behavior (see also --data):\n"
<< "    console width (set via _XCOLS macro)   : " << xeona::consoleWidth     << "\n"

<< "\n"

<< "command-line examples:\n"

<< "  run using all defaults including the default model name '"
<< xeona::modelStubDefault << xeona::modelExt
<< "':\n"
<< "    $ " << getProgramName() << "\n"
<< "  generate and run the compiled-in test model using the default name '"
<< xeona::modelInbuiltDefault << xeona::modelExt
<< "':\n"
<< "    $ " << getProgramName() << "  --inbuilt 24\n"
<< "  run '"
<< "mymodel" << xeona::modelExt
<< "' in restricted mode five and silent except for 'main.cc' log calls:\n"
<< "    $ " << getProgramName() << "  --mode 5  --report 0  --watch main  mymodel\n"
<< "  run '"
<< "mymodel" << xeona::modelExt
<< "' without defensive coding to completion or failure (and core dump):\n"
<< "    $ " << getProgramName() << "  --krazy  mymodel\n"
<< "  copy the guard model "
<< "'trial" << xeona::modelGuardTag << xeona::modelExt
<< "' to "
<< "'trial" << xeona::modelExt
<< "' first and run that:\n"
<< "    $ " << getProgramName() << "  --guard  trial\n"
<< "  run with enough reporting to track simulation progress:\n"
<< "    $ " << getProgramName() << "  --report 2  --watch overseer  --inbuilt 24\n"
<< "  reformat your 'untidy" << xeona::modelExt << "' model file"
<< " without adding entity output (shallow write):\n"
<< "    $ " << getProgramName() << "  --tidy  untidy\n"
<< "  minimal run which generates a representative two-step model file '"
<< xeona::modelInbuiltDefault << xeona::modelExt
<< "':\n"
<< "    $ " << getProgramName() << "  --mode 1  --inbuilt 2\n"
<< "  run using all defaults, but exit "
<< logga::kill << " if kill calls occur and exit " << logga::warn
<< " if warn calls occur:\n"
<< "    $ " << getProgramName() << "  --exittrip " << logga::warn << "\n"
<< "  minimal run which displays useful splash-screen information:\n"
<< "    $ " << getProgramName() << "  --report 0  --mode 1\n"
<< "  display all supported entity names, along with their respective headers:\n"
<< "    $ " << getProgramName() << "  --class +\n"
<< "  hunt for a particular entity name using the 'grep' pattern matching utility:\n"
<< "    $ " << getProgramName() << "  --class +  |  grep  --ignore-case  substring\n"
<< "  convenient way to read usage message using the 'less' pager:\n"
<< "    $ " << getProgramName() << "  --usage  |  less\n"
<< "  convenient way to read and search run-time output"
    " using the 'less' pager:\n"
<< "    $ " << getProgramName() << "  --report 6  mymodel  2>&1  |  less [+/WARN]\n"

<< "\n"

<< "entity name prefixes, useful with the --class option regex (can add leading ^):\n"
<< "    Asop = asset operator\n"
<< "    Cm   = commodity\n"
<< "    Cx   = context\n"
<< "    Gate = gateway\n"
<< "    Teas = technical asset\n"

<< "\n"

<< "subdirectory roles, useful with the --watch option:\n"
<< "    a/ = logging and reporting and xeona exceptions\n"
<< "    b/ = entity library excluding contexts\n"
<< "    c/ = workhorse code\n"
<< "    d/ = solver interface\n"
<< "    e/ = context library\n"
<< "    f/ = capset and transolve algorithm and similar\n"

<< "\n"
```

```
<< "option --yeek values (as currently databased):\n"
<< xeona::yeekSummarize(4)                  // note the indent argument

<< "\n"

<< "exit codes generated by the application:\n"
<< exitstatii                               // string from beginning of function block

<< "\n"

<< "exit codes due to operating system POSIX signals and some common causes:\n"
<< "   130 = user-initiated ^C (^ = ctrl) interrupt (not trapped)  SIGINT  =  2\n"
<< "   134 = glibc detects memory corruption during malloc call    SIGABRT =  6\n"
<< "   134 = glibc detects invalid pointer during free call        SIGABRT =  6\n"
<< "   134 = terminate on uncaught throw                           SIGABRT =  6\n"
<< "   134 = GLPK abort call                                       SIGABRT =  6\n"
<< "   137 = user-initiated hard kill (not trapped)                SIGKILL =  9\n"
<< "   139 = invalid memory use causing a segmentation fault       SIGSEGV = 11\n"
<< "   143 = user-initiated mild kill (not trapped)                SIGTERM = 15\n"

<< "\n"

<< "useful terminal control keys (else see $ stty --all)\n"
<< "    ^C  interrupt             not trapped                       SIGINT  =  2\n"
<< "    ^\\  quit                  not trapped, core dump            SIGQUIT =  3\n"
<< "    ^S  stop     ^Q  start\n"
<< "    ^Z  suspend  $ fg         pending output lost\n"

<< "\n"

<< "useful external programs (both have manpages):\n"
<< "  display resource usage information on completion (the options are correct)\n"
<< "    $ /usr/bin/time  --quiet  --verbose  " << getProgramName() << "\n"
<< "  check for memory leaks and faults (requires valgrind package)\n"
<< "    $ valgrind  --leak-check=full  " << getProgramName() << "\n"

<< "\n"

<< "command-line processing caution:\n"
<< "  options processed in the order given by --help and NOT in the order submitted\n"

<< "\n"

<< "software: "
<< getTitle()
<< "\n"

<< "notice: "
<< "distributed under the GNU General Public License version 3 and\n"
<< "  without warranty to the extent permitted by applicable law"
<< "\n"

<< "copyright: (c) "
<< getCopyright()
<< "\n";

  // note that Robbins (2005 pp255-300) devotes a chapter to POSIX signals

  s_logger->addSmartBlank();                // next logger call should add a blank line
  s_logger->addFinalStdoutBlank();          // add a final blank to stdout

  return ss.str();
}

const std::string
AppInfo::showLegalMessage() const
{
  std::ostringstream buf;
  buf
<< ""
<< "  Software  : " << getTitle()                                              << "\n"
<< ""                                                                          << "\n"
<< "  Copyright : (c) " << getCopyright() << "."                               << "\n"
<< ""                                                                          << "\n"
<< "  Request   : This software is distributed with the request that you forward" << "\n"
<< "              any modifications you make to the xeona project for possible"   << "\n"
<< "              inclusion in the main codebase."                             << "\n"
<< ""                                                                          << "\n"
<< "  License   : This software is distributed under the license which follows."  << "\n";
```

```
  std::ostringstream ss;
  ss << ""                              << "\n"
     << buf.str()                       << "\n"
     << ::licenseText                   << "\n"    // file-local string literal
     << "  End of xeona legal notice." << "\n"
     << ""                              << "\n";

  return ss.str();
}

const std::string
AppInfo::showXemRules() const
{
  std::string output;                       // variable to be returned
  output.reserve(10000);                    // estimate via $ xeona  --data  | wc --chars

  output +=                                 // intro line

    "\n"
    "xeona data rules\n";

  output +=                                 // rules

    "\n"

    "text alignment\n"
    "    column 0 has no leading space, column 4 has four leading spaces\n"
    "    record identifiers must start in column 0, data fields and comments may "
    "start anywhere\n"
    "    the same alignment rule applies to the record and field disable character "
    "'" + xeona::modelDisableChar + "'\n"
    "    by convention, data fields are placed in column 4 and comments in column 6\n"
    "\n"

    "xeona model file\n"
    "    a XEM file is a set of records\n"
    "    the record order has no semantic significance\n"
    "\n"

    "records\n"
    "    each record begins with an identifier placed in column 0 (otherwise the line is "
    "deemed a comment!)\n"
    "    the leading dot-separated part of each identifier indicates the "
    "record kind (see below)\n"
    "    processing ceases when the special '" + xeona::modelEndMarker + "' "
    "record identifier is met\n"
    "    blank lines are not intrinsically significant\n"
    "    column alignment is not significant, except for the placement of the "
    "record identifier\n"
    "    a note record is treated as set of comment lines, with multiple blank "
    "lines removed on output\n"
    "    records are written out in the order that they are read in\n"
    "\n"

    "record kinds (with identifiers):\n"
    "    note         'note'         : verbatim text "
    "(except multiple blank lines are squeezed)\n"
    "    program      'program.*'    : run information\n"
    "    entity       'entity.*'     : model entities\n"
    "    model end    'model-end'    : processing stops (remaining text is lost)\n"
    "\n"

    "fields\n"
    "    a record is made up of zero or more data fields and/or comments\n"
    "    any field containing a single angle bracket is a data field, otherwise it "
    "is a comment\n"
    "    each data field has a leading identifier, followed optionally by \"[unit]\" "
    "and optionally by a remark\n"
    "    a comment may possess matched angle brackets (for instance, "
    "<email@address> is legal in a comment)\n"
    "\n"

    "data\n"
    "    a data field may hold a single value, a timeseries, an individual "
    "entity identifier, or a list of identifiers\n"
    "    single or timeseries values may be of data-type "
    "string, integer, float, or boolean\n"
    "    a string value is set in double quotes, a boolean value is either 0 (false) "
    "or 1 (true)\n"
    "    a connectivity list comprises zero or more space-separated entity "
    "identifiers placed within one set of double quotes\n"
```

```
    "    entities may disallow the use of null entity identifiers in some cases "
    "(as documented)\n"
    "\n"

    "disabling records and fields\n"
    "    a record can be disabled by placing a leading '" + xeona::modelDisableChar + "'"
    " in column 0 (and nowhere else)\n"
    "    a data field can be disabled by placing a leading "
    "'" + xeona::modelDisableChar + "' anywhere (but best to avoid column 0)\n"
    "    a disabled field must appear AFTER its respective enabled field\n"
    "    a leading '" + xeona::modelDisableChar + "' on a comment line is "
    "automatically removed and not replaced (think of this as a feature)\n"
    "\n"

    "reformatting on write out\n"
    "    xeona reformats the XEM file on write out\n"
    "    most notably, in-data, out-data, and comments are grouped together "
    "in that order\n"
    "    additional support is provided for emacs text editor local variables "
    "if originally present\n"
    "\n"

    "field kinds:\n"
    "    comment     : no angle bracket (or alternatively matched angle brackets)\n"
    "    in-data     : >\n"
    "    out-data    : <\n"
    "\n"

    "data values\n"
    "    floats may be input in either decimal or exponent format: "
    "-0.01234 or -12.34e-03\n"
    "    a trailing \"" + xeona::modelTsRepeater + "\" on timeseries input data "
    "means repeat that pattern as often as needed\n"
    "    xeona holds floats using the 8-byte 'double' arithmetic type"
    " (assuming IEEE 754 support)\n"
    "\n"

    "mandatory entities with preset identifiers:\n"
    "    TimeHorizon   entity.time-horizon\n"
    "    Overseer      entity.overseer\n";

  output +=                                 // field value prompts

    "\n"
    "field value data-type prompts "
    "(not actively parsed but provided to assist data preparation):\n"
    "        0  null for completeness\n"
    "    entity connectivity (in-data only, both can be null in some circumstances)\n"
    "        l  individual entity identifier within double quotes\n"
    "        L  space-separated list of entity identifiers within one set of "
    "double quotes\n"
    "    normal data (both in-data and out-data)\n"
    "        s  single string in double quotes\n"
    "        S  timeseries of strings with each element in double quotes\n"
    "        i  single integer\n"
    "        I  timeseries of integers\n"
    "        f  single floating-point number\n"
    "        F  timeseries of floating-point numbers\n"
    "        b  single boolean\n"
    "        B  timeseries of booleans\n"
    "    reinterpreted data (entity-specific in-data)\n"
    "        x  single character stream within double quotes "
    "(for instance, bidset data)\n"
    "        X  timeseries of above with each element in double quotes\n";

  output +=                                 // field units

    "\n"
    "field units (enclosed in square brackets):\n"
    "    physical quantities: unprefixed SI units are presumed "
    "(unless otherwise documented)\n"
    "        time: s\n"
    "        raw SI units: kg m ohm V etc\n"
    "        derived SI units: J Pa W etc\n"
    "        temperature: C or K (C preferred unless calculation utilizes K)\n"
    "    economic quantities:\n"
    "        time: y (year can be more intuitive than second in certain cases, "
    "for instance, rate of interest)\n"
    "        generic currency: $ (representing, for example, EUR or USD)\n"
    "    quantities lacking dimensionality:\n"
    "        unitless: - (normally preferred over %)\n"
```

```
    "        percentage: %\n"
    "    embedded costs:\n"
    "        reported $ per-interval (not $/s)\n";

  output +=                                 // entity documentation

    "\n"
    "entity documentation:\n"
    "    entity documentation can be obtained through xeona option --class +|regex\n"
    "    commodity-templated entities often use generic documentation, such that:\n"
    "        base commodity in { Cert, Cseq, Elec, Fiss, Fund, Heat, Oxid, Thrm, Work }\n"
    "        stand-in quantifying extensity field"
    " unit: * in { kg, J, $ } as appropriate\n";

  output +=                                 // compiled-in literals

    "\n"
    "compiled-in literals relating to data usage (set in unit 'common'):\n"
    "    ISO 4217 currency code                : " + xeona::iso4217          + "\n"
    "    record or field disable string        : " + xeona::modelDisableChar + "\n"
    "    timeseries repeat indicator           : " + xeona::modelTsRepeater  + "\n"
    "    pairwise delimiter for string values  : " + xeona::modelStringDelim + "\n"
    "    nodal bid and tariff entry separator  : " + xeona::modelBidDelim    + "\n"
    "    model end marker                      : " + xeona::modelEndMarker   + "\n"
    "    separator string for OSP tag reporting : " + xeona::ospTagSep        + "\n";

  output +=                                 // abbreviations

    "\n"
    "data-related abbreviations:\n"
    "    EUR     euro\n"
    "    OSP     optimization sub-problem\n"
    "    XEM     xeona model\n";

  const std::string trial    = "trial";
  const std::string trialxem = trial + xeona::modelGuardTag + xeona::modelExt;

  output +=                                 // useful command-lines

    "\n"
    "some useful command-lines for developing XEM files:\n"
    "    $ " + getProgramName() + "  --inbuilt 2\n"
    "    $ cp  " + xeona::modelInbuiltDefault + xeona::modelExt + xeona::backupTag + "  "
    + trialxem + "\n"
    "    $ emacs  " + trialxem + "  &    # or your preferred editor\n"
    "    $ " + getProgramName() + "  --guard " + trial + "  "
    "&&  cat  " + trial + xeona::modelExt + "\n";

  output +=                                 // final caution

    "\n"
    "final caution:\n"
    "    entity authors may not have respected these guidelines in their entirety\n"
    "    that said, non-standard practice should be signaled via the"
    " builtin-remark field\n";

  output += "\n";                           // blank line

  // return the output
  return output;
}

const std::string
AppInfo::showParse() const
{
  std::string FS = " : ";                   // field separator

  std::string debugInfo                = "0";
  if ( xeona::DBUG ) debugInfo         = "1";

  std::string ndebugInfo               = "0";
  if ( xeona::NBUG ) ndebugInfo        = "1";

  std::string buildStatus              = "non-release";
  if ( getReleaseStatus() ) buildStatus = "release";

  std::ostringstream ss;
  ss << "invoked-name"
     << FS
     << getProgramName()
     << "\n"
```

```
      << "compile-date"
      << FS
      << getBuildDate()
      << "\n"

      << "build-status"
      << FS
      << buildStatus
      << "\n"

      << "codebase-status"
      << FS
      << getCodebaseStatus()
      << "\n"

      << "_XDEBUG-app"
      << FS
      << debugInfo
      << "\n"

      << "NDEBUG-libs"
      << FS
      << ndebugInfo
      << "\n"

      << "svn-revision"
      << FS
      << getSvnRevision()
      << "\n";

  return ss.str();
}

// Note: representative output and script parsing examples using
// awk (also gawk, nawk) AND "begin" should read "BEGIN" (duly
// changed to prevent '/usr/bin/file' from misreporting this
// file)
//
// $ xeona-mach --version
// invoked-name : xeona-mach
// code-status : work-in-progress
// debug-info : omitted
// svn-date : 2007-04-17
// svn-revision : 347
//
// $xeona-mach --version [backslash]
//   | awk 'begin { FS = " : " } $1 ~ /svn-revision/ { print $2 }'
// 10
// note that the default RS is a newline

const std::string
AppInfo::showFinal
(xeona::SimRet simRet,
 const int     exitStatus) const
{
  // process logger rank for this stream
  const std::string logCallList = s_logger->getAllTriggerStr();

  // process xeona::SimRet enum
  std::string simInterp = "(string not overwritten)";
  switch ( simRet )
    {
    case xeona::e_statusNotKnown:   simInterp = "indeterminate";                break;
    case xeona::e_success:          simInterp = "simulate() returned success";  break;
    case xeona::e_modelFileFault:   simInterp = "faulty model file";            break;
    case xeona::e_infeasibility:    simInterp = "infeasibility encountered";    break;
    case xeona::e_errantSimulation: simInterp = "problems encountered";         break;
    case xeona::e_testCodeUsed:     simInterp = "downrated test code utilized"; break;
    case xeona::e_other:            simInterp = "(other)";                      break;
    default:
      s_logger->repx(logga::warn, "uncoded simRet", simRet);
    }

  // stop the clock!
  //
  // There is no real loss of accuracy having the application
  // timer located within the 'AppInfo' class rather than within
  // some global class.  A full run with a non-optimized binary,
  // with _XDEBUG set, and an deliberately incorrect model file
  // name takes under 10 milliseconds on my circa 2004 32-bit
```

```
  // laptop (gogol) with 1.4GHz Intel Celeron M processor.
  // Hence:
  //
  //     $ /usr/bin/time --portability xeona.mach --report 0 xxx
  //
  //     Command exited with non-zero status 10
  //     real 0.01
  //     user 0.00
  //     sys 0.00

  boost::posix_time::ptime toc(boost::posix_time::microsec_clock::universal_time());
  boost::posix_time::time_duration delta = toc - d_tic;

  std::string exitInterp;
  {                                       // local block to force early dtor logging
    ExitStatus exitstatus("(coding error, rerun at --report 2)"); // not-present message
    if ( ! exitstatus(exitStatus, exitInterp) )
      {
        s_logger->repx(logga::warn, "exit status not in database", exitStatus);
      }
  }

  // prepare the output
  std::ostringstream ss;
  ss << ""                                                                  << "\n"
     << "  final information"                                               << "\n"
     << ""                                                                  << "\n"
     << "     command-line      :  " << d_fullCmdLine                       << "\n"
     << "     start directory   :  " << d_startPath                         << "\n"
     << "     elapsed time      :  " << xeona::formatDuration(delta)        << "\n"
     << "     log calls to date :  " << logCallList                         << "\n"
     << "     simulation state  :  " << simInterp                           << "\n"
     << "     application exit  :  " << exitStatus << " = " << exitInterp   << "\n";

  s_logger->addSmartBlank();                // next logger call should add a blank line
  s_logger->addFinalStdoutBlank();          // add a final blank to stdout

  return ss.str();
}

const std::string
AppInfo::getLogname() const
{
  const char* logname = getenv("LOGNAME");   // <cstdlib>
  if ( logname ) return logname;
  else           return "(not determined)";
}

const std::string                           // format: 2007-04-16
AppInfo::getCurrentDate() const
{
  boost::posix_time::ptime now
    = boost::posix_time::second_clock::universal_time();
  return boost::gregorian::to_iso_extended_string(now.date());
}

const std::string                           // format: 23:59:59
AppInfo::getCurrentTime() const
{
  boost::posix_time::ptime now
    = boost::posix_time::second_clock::universal_time();
  return boost::posix_time::to_simple_string(now.time_of_day());
}

const std::string
AppInfo::getReportLevel() const
{
  std::ostringstream ss;
  ss << s_logger->getReportLevelStr()
     << " ("
     << s_logger->getReportLevelInt()
     << ")";
  return ss.str();
}

const std::string
AppInfo::getCommandLineModelName() const
{
  return d_clModelName;
}
```

```
// MANIPULATORS

void
AppInfo::setCommandLineReportLevel
(const unsigned reportLevel)
{
  d_clReportLevel = reportLevel;
}

void
AppInfo::setCommandLineModelName
(const std::string& modelName)
{
  d_clModelName = modelName;
}

// PRIVATE FUNCTIONS

bool                                        // fail also sets field to ""
AppInfo::splitString
(const std::string& record,                 // input space-separated string
 std::string&       field,                  // selected output field
 const unsigned int index) const            // field index, zero-based
{
  std::vector<std::string> output;                      // buffer
  boost::split(output, record, boost::is_any_of(" ")); // useful 'split' function
  if (index < output.size())
    {
      field = output.at(index);             // will throw if out of range
      return true;
    }
  else
    {
      field = "";
      return false;
    }
}

const boost::filesystem::path
AppInfo::pathFromString
(const std::string& path) const
{
  // create buffer string
  std::string buff(path);                   // CAUTION: also converts to non-const

  // strip any leading "./" or ".\\"
  const char native = xeona::osSlash;       // native path separator set in 'common.cc'
  const std::string slash(1, native);       // CAUTION: note char constructor syntax
  const std::string dot(1, '.');
  const std::string junk = dot + slash;     // the offending non-leaf path
  if ( boost::starts_with(buff, junk) )
    buff = buff.substr(2);                  // remove first two chars

  // form complete (absolute) path
  return boost::filesystem::absolute(buff); // note the default second argument [1]

  // [1] boost::filesystem::absolute() has a default second
  // argument of boost::filesystem::initial_path(), this being
  // the current directory at the time of entry into main().
  //
  // CAUTION: system_complete has different behavior so read the
  // documentation if you wish to change from POSIX conformance
  // to operating system-based format rules
}

const std::string
AppInfo::extractLeaf                        // the leaf is the final part of the path
(const boost::filesystem::path path) const
{
  boost::filesystem::path temp
    = d_progPath.filename();                // grab leaf
  return temp.string();                     // 'file_string' outputs os native format
}

const std::string
AppInfo::extractPath
(const boost::filesystem::path path) const
{
  return d_progPath.string();               // 'file_string' outputs os native format
}
```

```
// pad numbers
std::string
AppInfo::zeroPadMe
(const std::string& number,
 const int         padlevel) const
{
  int num = boost::lexical_cast<int>(number);
  std::ostringstream ossBuf;                  // formatting buffer
  ossBuf << std::setw(padlevel)               // define overall width
         << std::setfill('0')                 // define zero pad character
         << std::fixed                        // adopt fixed formatting
         << num;
  return ossBuf.str();
}

std::string
AppInfo::argProcess
(int    argc,
 char** argv) const
{
  std::vector<std::string> argValues;
  std::string argConcat;

  for ( int i = 0; i < argc; ++i )
    argValues.push_back(argv[i]);

  BOOST_FOREACH( std::string s, argValues )
    argConcat += s + " ";

  boost::trim(argConcat);                     // trim trailing space
  return argConcat;
}

// STATIC DATA

logga::spLogger
AppInfo::s_logger = logga::ptrLogStream();   // bind logger on definition

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : app_except.cc
//  file-create-date : Tue 28-Apr-2009 10:15 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : application exception classes / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exapp.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "exapp.h"            // companion header for this file (place first)

#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting

//  CODE

namespace xeona
{
  // ------------------------------------------------------------
  //  CLASS           : xeona::app_exception (abstract)
  // ------------------------------------------------------------

  app_exception::app_exception
  (const int exitcode) :
    exception(exitcode)
  { }

  app_exception::~app_exception()
  { }                                       // definition necessary

  // ------------------------------------------------------------
  //  CLASS           : xeona::kill_on_log
  // ------------------------------------------------------------

  kill_on_log::kill_on_log() :
    app_exception(xeona::exit_kill_on_log)
  {
    d_stringExpl = "";                      // no newline is correct
    d_stringTell = "xeona::kill_on_log exception";

  } // 'xeona::kill_on_log'

  // ------------------------------------------------------------
  //  CLASS           : xeona::non_registration
```

```
// ------------------------------------------------------

non_registration::non_registration
(const std::string entityIdentifier,       // 'entityId'
 const std::string entityRegistration,     // 'entityRegn'
 const std::string requestedClass) :       // 'r.locateClass()'
   app_exception(xeona::exit_non_registration)
{
  std::ostringstream oss;
  oss
    << "** xeona non-registration exception"                                  << "\n"
    << "   unregistered entity class requested"                               << "\n"
    << "       entity identifier   : " << entityIdentifier                    << "\n"
    << "       requested class     : " << requestedClass                      << "\n"
    << "       entity registration : " << entityRegistration                  << "\n"
    << "   likely causes"                                                     << "\n"
    << "       modeler misspelled entity class name in model file"            << "\n"
    << "       entity author failed to register entity class properly"        << "\n"
    << "   hardcoded exit status : " << d_code                                << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::non_registration exception";

} // 'xeona::non_registration'

// ------------------------------------------------------
//  CLASS           : xeona::empty_wrap
// ------------------------------------------------------

empty_wrap::empty_wrap
(const std::string type) :                 // 'xeona::demangle(typeid(T).name())'
   app_exception(xeona::exit_empty_wrap)
{
  std::ostringstream oss;
  oss
    << "** xeona empty wrap exception"                                        << "\n"
    << "   about to assign from an empty (default constructed) wrap<> object" << "\n"
    << "       parameterization : " << type                                   << "\n"
    << "   likely causes"                                                     << "\n"
    << "       input  : mismatch of supplied and required model data"         << "\n"
    << "       output : entity failed to produce suitably structured output"  << "\n"
    << "   for more information (including field name) see recent logging"    << "\n"
    << "   also try --class CLASSNAME for required model data format"         << "\n"
    << "   hardcoded exit status : " << d_code                                << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::empty_wrap exception";

} // 'xeona::empty_wrap'

// ------------------------------------------------------
//  CLASS           : xeona::short_timeseries
// ------------------------------------------------------

short_timeseries::short_timeseries
(const std::string fieldName,              // 'getName()'
 const int         shortfall) :            // 'int delta = horizon - len'
   app_exception(xeona::exit_short_timeseries)
{
  std::ostringstream oss;
  oss
    << "** xeona short timeseries exception"                                  << "\n"
    << "   short timeseries encountered"                                      << "\n"
    << "       field name : " << fieldName                                    << "\n"
    << "       shortfall  : " << shortfall                                    << "\n"
    << "   likely causes"                                                     << "\n"
    << "       insufficient data (try adding trailing "
    << xeona::modelTsRepeater << ")"                                          << "\n"
    << "   for more information see recent logging output"                    << "\n"
    << "   hardcoded exit status : " << d_code                                << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::short_timeseries exception";

} // 'xeona::short_timeseries'

// ------------------------------------------------------
//  CLASS           : xeona::file_not_found
// ------------------------------------------------------

file_not_found::file_not_found
```

```
(const std::string filename,
 const std::string comment) :
  app_exception(xeona::exit_file_not_found)
{
  std::ostringstream oss;
  oss
    << "** xeona file not found exception"                                    << "\n"
    << "       file name : " << filename                                     << "\n"
    << "         comment : " << comment                                      << "\n"
    << "   hardcoded exit status : " << d_code                               << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::file_not_found exception";

} // 'xeona::file_not_found'

// ----------------------------------------------------------
//  CLASS           : xeona::xem_data_issue
// ----------------------------------------------------------

xem_data_issue::xem_data_issue
(const std::string comment,
 const std::string name,
 const std::string valueStr) :
  app_exception(xeona::exit_xem_data_issue)
{
  // 'value' can be 1000s of chars long so it needs truncating
  std::string trunc(valueStr);
  const unsigned width    = xeona::consoleWidth;  // often set to 142
  const unsigned strwidth = width - 20;           // CAUTION: unsigned for comparison
  if ( trunc.length() > strwidth ) trunc.erase(strwidth);
  boost::trim(trunc);                     // trim trailing space

  std::ostringstream oss;
  oss
    << "** xeona xem data issue exception"                                    << "\n"
    << "         comment : " << comment                                      << "\n"
    << "            name : " << name                                         << "\n"
    << "    value string : " << trunc                                        << "\n"
    << "    value length : " << valueStr.length()                            << "\n"
    << "   hardcoded exit status : " << d_code                               << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::xem_data_issue exception";

} // 'xeona::xem_data_issue'

// ----------------------------------------------------------
//  CLASS           : xeona::lazy_link_fail
// ----------------------------------------------------------

lazy_link_fail::lazy_link_fail
(const std::string comment,
 const std::string linkname) :
  app_exception(xeona::exit_lazy_link_fail)
{
  std::ostringstream oss;
  oss
    << "** xeona lazy link fail exception"                                    << "\n"
    << "         comment : " << comment                                      << "\n"
    << "       link name : " << linkname                                     << "\n"
    << "   hardcoded exit status : " << d_code                               << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::lazy_link_fail exception";

} // 'xeona::lazy_link_fail'

// ----------------------------------------------------------
//  CLASS           : xeona::cannot_run_guard
// ----------------------------------------------------------

cannot_run_guard::cannot_run_guard
(const std::string filename) :
  app_exception(xeona::exit_cannot_run_guard)
{
  std::ostringstream oss;
  oss
    << "** xeona cannot run guard file exception"                             << "\n"
    << "       file name : " << filename                                     << "\n"
    << "   likely fix"                                                       << "\n"
```

```
      << "       add option --guard to the command-line and rerun"               << "\n"
      << "   hardcoded exit status : " << d_code                                 << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::cannot_run_file exception";

} // 'xeona::cannot_run_guard'

// ------------------------------------------------------
//  CLASS           : xeona::bad_authorship
// ------------------------------------------------------

bad_authorship::bad_authorship
(const int numberFailedEntities) :
  app_exception(xeona::exit_bad_authorship)
{
  std::ostringstream oss;
  oss
    << "** xeona bad authorship exception"                                      << "\n"
    << "       number of failed entities : " << numberFailedEntities           << "\n"
    << "   likely fix"                                                         << "\n"
    << "       correct the problematic entities"                               << "\n"
    << "   hardcoded exit status : " << d_code                                 << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::bad_authorship exception";

} // xeona::bad_authorship'

// ------------------------------------------------------
//  CLASS           : xeona::full_link_fail
// ------------------------------------------------------

full_link_fail::full_link_fail
(const std::string identifier,
 const std::string etype,
 const std::string mtype) :
  app_exception(xeona::exit_full_link_fail)
{
  std::ostringstream oss;
  oss
    << "** xeona full link entity (information flows) fail exception"          << "\n"
    << "       identifier       : " << identifier                               << "\n"
    << "       cast (sub-)type  : " << etype                                    << "\n"
    << "       my type          : " << mtype                                    << "\n"
    << "   likely fix"                                                         << "\n"
    << "       mismatch of supplied and required model data"                   << "\n"
    << "       link problems include (but are not limited to) context entities" << "\n";
  if ( identifier.empty() )
    {
      oss << "       in this case, provide a non-empty link identifier"        << "\n";
    }
  else
    {
      oss << "       in this case, check presence of named link as an entity"  << "\n";
    }
  oss
    << "   hardcoded exit status : " << d_code                                 << "\n";

  d_stringExpl = oss.str();
  d_stringTell = "xeona::full_link_fail exception";
}

// ------------------------------------------------------
//  CLASS           : xeona::empty_field_on_write
// ------------------------------------------------------

empty_field_on_write::empty_field_on_write
(const std::string fieldType,
 const std::string fieldname) :
  app_exception(xeona::exit_empty_field_on_write)
{
  std::ostringstream oss;
  oss
    << "** xeona empty field on write out exception"                            << "\n"
    << "       field type : " << fieldType                                     << "\n"
    << "       field name : " << fieldname                                     << "\n"
    << "   likely fix"                                                         << "\n"
    << "       mismatch of supplied and required field name"                   << "\n"
    << "       this problem should have also produced prior warnings"          << "\n"
    << "   hardcoded exit status : " << d_code                                 << "\n";
```

```
    d_stringExpl = oss.str();
    d_stringTell = "xeona::empty_field_on_write exception";

  } // 'xeona::empty_field_on_write'

  // ------------------------------------------------------
  //  CLASS           : xeona::timeseries_not_found
  // ------------------------------------------------------

  timeseries_not_found::timeseries_not_found
  (const std::string fieldname,
   const std::string templateType) :
    app_exception(xeona::exit_timeseries_not_found)
  {
    // process information
    const std::string utype = "shared_ptr<vector<" + templateType + "> >";
    std::string       fvdt  = "(not set)";           // field value data-type
    if      ( templateType == "bool"   ) fvdt = "B";
    else if ( templateType == "int"    ) fvdt = "I";
    else if ( templateType == "double" ) fvdt = "F";
    else if ( templateType == "string" ) fvdt = "S,L,X";    // CAUTION: not "std::string"

    std::ostringstream oss;
    oss
      << "** xeona timeseries not found exception"                              << "\n"
      << "       sought field name : " << fieldname                             << "\n"
      << "       underlying type   : " << utype                                 << "\n"
      << "       interpretation    : " << fvdt                                  << "\n"
      << "   likely fix"                                                        << "\n"
      << "       mismatch of supplied and required field name"                  << "\n"
      << "       this problem should have also produced prior warnings"         << "\n"
      << "       check previous reporting for associated entity identifier"     << "\n"
      << "   hardcoded exit status : " << d_code                                << "\n";

    d_stringExpl = oss.str();
    d_stringTell = "xeona::timeseries_not_found exception";

#if 1

    // CAUTION: for some (completely mysterious) reason, this
    // exception was not being caught in 'main', so report
    // directly at construction-time in anticipation of a later
    // forced abort

    std::clog << std::flush
              << "\n"
              << " ++++ special construction-time report because"
              << " this exception was not being caught ++++" << "\n"
              << "\n"
              << d_stringExpl
              << std::flush;

    std::cout << std::flush
              << "\n"
              << "** " << d_stringTell << "\n"
              << "\n"
              << std::flush;

#endif // 0

  } // 'xeona::timeseries_not_found'

  // ------------------------------------------------------
  //  CLASS           : xeona::yeek_abandon
  // ------------------------------------------------------

  yeek_abandon::yeek_abandon
  (const std::string explanation) :
    app_exception(xeona::exit_yeek_abandon)
  {
    std::ostringstream oss;
    oss
      << "** xeona yeek abandon exception"                                      << "\n"
      << "       yeek value  : " << xeona::yeek                                << "\n"
      << "       explanation : " << explanation                                << "\n"
      << "   purpose"                                                          << "\n"
      << "       this exception does not indicate a fault state"               << "\n"
      << "       it is used to trip the application under supported yeek values"  << "\n"
      << "   hardcoded exit status : " << d_code                               << "\n";
```

```
    d_stringExpl = oss.str();
    d_stringTell = "xeona::yeek_abandon exception";

  } // 'xeona::yeek_abandon'

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : exbase.cc
//  file-create-date : Wed 24-Jun-2009 05:49 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : xeona exception base class / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exbase.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "exbase.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

namespace xeona
{
  // ------------------------------------------------------------
  //  CLASS           : xeona::exception (abstract base)
  // ------------------------------------------------------------

  exception::exception
  (const int exitcode) :
    d_code(exitcode),
    d_stringExpl(),
    d_stringTell()
  {
#if 1 // 1 = flush ostreams on construction, 0 = omit
    std::cout << std::flush;
    std::clog << std::flush;
#endif // 0
  }

  const std::string exception::expl() const { return d_stringExpl; }
  const std::string exception::tell() const { return d_stringTell; }
  const int         exception::code() const { return d_code;       }

  exception::~exception() { }               // definition necessary

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : app_except.cc
//  file-create-date : Tue 28-Apr-2009 10:15 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : entity exception classes / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exent.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "exent.h"            // companion header for this file (place first)

#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting

//  CODE

namespace xeona
{
  // ------------------------------------------------------------
  //  CLASS           : xeona::ent_exception (abstract)
  // ------------------------------------------------------------

  ent_exception::ent_exception() :
    exception(xeona::exit_entity_fail)       // only this exit code is supported
  {
  }

  ent_exception::~ent_exception()            // definition necessary
  {
  }

  // ------------------------------------------------------------
  //  CLASS           : xeona::entity_issue
  // ------------------------------------------------------------

  entity_issue::entity_issue
  (const std::string msgExpl)      // should be suitably formatted with trailing newline
  {
    std::ostringstream oss;
    oss
      << "** xeona entity issue exception"                                  << "\n"
      << "   "                          << msgExpl                         << "\n"
      << "   hardcoded exit status : " << d_code                           << "\n";
```

```
    d_stringExpl = oss.str();
    d_stringTell = "xeona::entity_issue exception";
  }

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : exitstat.cc
//  file-create-date : Thu 14-May-2009 07:11 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : exit status database / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/exitstat.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "exitstat.h"         // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <numeric>            // STL numerical algorithms
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <utility>            // STL pair, make_pair()

//  CODE

//  STATIC DEFINITIONS

logga::spLogger ExitStatus::s_logger = logga::ptrLogStream();

// ---------------------------------------------------------
//  MEMBER FUNCTION : ExitStatus
// ---------------------------------------------------------

// COMMENT: modify option '--output' text here

ExitStatus::ExitStatus
(const std::string notPresentMessage) :
  d_data(),                                 // empty map
  d_notPresentMessage(notPresentMessage)
{
  s_logger->repx(logga::dbug, "constructor call", "");

  // data load on construction (can break string lines with intervening " pair)

  add(
      //  0
      xeona::ret_success,
      "success (relative to prevailing exit trip, the default is to accept warnings)"
      );
  add(
      //  1
      logga::kill,
```

```
    "kill report calls made (requires exit trip 1 or greater)"
    );
add(
    //  2
    logga::warn,
    "warning report calls made (requires exit trip 2 or greater)"
    );
add(
    //  3
    logga::info,
    "information report calls made (requires exit trip 3 or greater)"
    );
add(
    //  4
    logga::dbug,
    "debug report calls made (requires exit trip 4 or greater)"
    );
add(
    //  5
    logga::xtra,
    "extra report calls made (requires exit trip 5 or greater)"
    );
add(
    //  6
    logga::adhc,
    "ad-hoc report calls made (requires exit trip 6)"
    );
add(
    //  8
    xeona::ret_noclass,
    "no match for given --class arg"
    );
add(
    //  9
    xeona::ret_usage,
    "command-line usage issue (refer to the displayed explanation)"
    );
add(
    // 10
    xeona::exit_kill_on_log,
    "xeona exception on printed KILL log call (where such action is enabled)"
    );
add(
    // 11
    xeona::exit_empty_wrap,
    "xeona exception on empty wrap object (check entity data/class match)"
    );
add(
    // 12
    xeona::exit_non_registration,
    "xeona exception on failure to find entity class name (check class string)"
    );
add(
    // 13
    xeona::exit_short_timeseries,
    "xeona exception on short timeseries (check data length vs time-horizon.steps)"
    );
add(
    // 14
    xeona::exit_file_not_found,
    "xeona exception on file not found"
    );
add(
    // 15
    xeona::exit_xem_data_issue,
    "xeona exception on faulty xem data (duplicate name or hanging association)"
    );
add(
    // 16
    xeona::exit_lazy_link_fail,
    "xeona exception on lazy link fail (check link name)"
    );
add(
    // 17
    xeona::exit_cannot_run_guard,
    "xeona exception on attempt to directly run guard file (use option --guard)"
    );
add(
    // 18
    xeona::exit_bad_authorship,
    "xeona exception on badly authored entities (coding problem)"
```

```
    );
add(
    // 19
    xeona::exit_full_link_fail,
    "xeona exception on full link entity fail (data problem)"
    );
add(
    // 20
    xeona::exit_empty_field_on_write,
    "xeona exception on empty field on write out (data problem)"
    );
add(
    // 21
    xeona::exit_timeseries_not_found,
    "xeona exception on missing timeseries field (data problem)"
    );
add(
    // 22
    xeona::exit_yeek_abandon,
    "xeona exception on optional application yeek code"
    );
add(
    // 30
    xeona::exit_entity_fail,
    "xeona exception thrown by entity author (read associated message)"
    );
add(
    // 40
    xeona::exit_bad_assign_cast,
    "xeona::assign_ptr cast failed"
    );
add(
    // 41
    xeona::exit_boost_exception,
    "unexpected boost library exception"
    );
add(
    // 42
    xeona::exit_std_out_of_range,
    "unexpected standard out-of-range exception"
    );
add(
    // 43
    xeona::exit_std_logic_error,
    "unexpected standard  logic error (read associated message)"
    );
add(
    // 44
    xeona::exit_std_bad_alloc,
    "insufficient heap (mid-simulation) memory"
    );
add(
    // 45
    xeona::exit_std_exception,
    "unexpected standard library exception"
    );
add(
    // 46
    xeona::exit_unknown_exception,
    "unexpected unknown exception"
    );
add(
    // 50
    xeona::ret_model_file_fault,
    "simulation call encountered faulty model file (may not be writable)"
    );
add(
    // 51
    xeona::ret_infeasibility,
    "simulation call encountered infeasibility (solver fail or problem choke)"
    );
add(
    // 52
    xeona::ret_errant_simulation,
    "simulation call returned non-differentiated fault status"
    );
add(
    // 53
    xeona::ret_test_code_used,
    "simulation call encountered test code within a release build (coding mistake)"
    );
```

```
  add(
      // 58
      xeona::ret_other,
      "simulation call returned other"
      );
  add(
      // 59
      xeona::ret_status_not_known,
      "simulation call returned unknown status (dummy value not overwritten)"
      );
  add(
      // 99
      xeona::exit_test99,
      "reserved for code development purposes"
      );

  // from here, supported under option '--output' but not '--usage'

  add(
      // 130
      130,
      "user-initiated ^C"
      );
  add(
      // 134
      134,
      "bad malloc or free / uncaught throw / GLPK abort"
      );
  add(
      // 139
      139,
      "segfault on invalid memory use"
      );
  add(
      // 255
      xeona::ret_fail,
      "coding error or similar"
      );
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~ExitStatus
// ------------------------------------------------------------

ExitStatus::~ExitStatus()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : operator() (bool return)
// ------------------------------------------------------------

bool                                        // return 'false' if not present
ExitStatus::operator()
(const int    exitStatus,
 std::string& exitInterpretation) const
{
  // Josuttis (1999 p212) describes the various search options
  database_type::const_iterator pos = d_data.find(exitStatus);
  if ( pos == d_data.end() )
    {
      exitInterpretation = d_notPresentMessage;
      return false;
    }
  else
    {
      exitInterpretation = pos->second;
      return true;
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : notPresentMessage
// ------------------------------------------------------------

std::string
ExitStatus::notPresentMessage() const
{
  return d_notPresentMessage;
}
```

```
// -----------------------------------------------------------
//  MEMBER FUNCTION : size
// -----------------------------------------------------------

int
ExitStatus::size() const
{
  return static_cast<int>(d_data.size());    // cast from 'std::map<>::size_type'
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : add
// -----------------------------------------------------------

void
ExitStatus::add
(const int         exitStatus,
 const std::string exitInterpretation)
{
  // Josuttis (1999 p203) describes the "second" insertion success test
  if ( ! d_data.insert(std::make_pair(exitStatus, exitInterpretation)).second )
    {
      s_logger->repx(logga::warn, "duplicate key insertion attempted", exitStatus);
    }
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : floatstat.cc
//  file-create-date : Mon 20-Jul-2009 10:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : floating-point environment management / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/floatstat.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "floatstat.h"        // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : xeona::SaveClearResetFeFlag
// ----------------------------------------------------------

namespace xeona
{
  // CREATORS

  SaveClearResetFeFlag::SaveClearResetFeFlag
  (const int feFlag) :                      // macros as defined in <fenv.h>, can be |'ed
    d_feFlag(feFlag),
    d_feStat()                              // default construction
  {
    fegetexceptflag(&d_feStat, d_feFlag);   // obtain status
    feclearexcept(d_feFlag);                // duly clear status
  }

  SaveClearResetFeFlag::~SaveClearResetFeFlag()
  {
    fesetexceptflag(&d_feStat, d_feFlag);   // restore call
  }

  // ACCESSORS

  bool
  SaveClearResetFeFlag::tripped() const
  {
    if ( fetestexcept(d_feFlag) ) return true;    // returns 'int', see manpage
    else                          return false;
```

```
  }

  int
  SaveClearResetFeFlag::getFlag() const
  {
    return d_feFlag;
  }

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : helpers.cc
//  file-create-date : Mon 18-May-2009 11:08 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : application helper functions / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/helpers.cc $
//
//  GENERAL NOTES FOR THIS FILE
//  LOCAL AND SYSTEM INCLUDES
#include "helpers.h"          // companion header for this file (place first)
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::logRankToGlpkLevel
// ----------------------------------------------------------

namespace xeona
{
  const svif::ReportingLevel
  logRankToGlpkLevel
  (const unsigned loggerRank)               // fundamentally 'logga::Rank'
  {
    // logging support
    static logga::spLogger logger = logga::ptrLogStream();
    logger->repx(logga::xtra, "entering member function", "");

    // declare return variable
    svif::ReportingLevel glpkLevel = svif::not_specified;

    // set the policy here
    switch ( loggerRank )
      {
      case logga::yeek: glpkLevel = svif::silent; break;
      case logga::kill: glpkLevel = svif::low;    break;
      case logga::warn: glpkLevel = svif::low;    break;
      case logga::info: glpkLevel = svif::high;   break;
      case logga::dbug: glpkLevel = svif::high;   break;
      case logga::xtra: glpkLevel = svif::high;   break;
      case logga::adhc: glpkLevel = svif::high;   break;
      default:
        std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
        break;
      }

    // for reporting purposes
```

```
    std::string glpkInterpretation;
    switch ( glpkLevel )
      {
      case svif::silent: glpkInterpretation = "no output (leakage may occur)"; break;
      case svif::low:    glpkInterpretation = "errors and warnings";           break;
      case svif::medium: glpkInterpretation = "normal (without added info)";   break;
      case svif::high:   glpkInterpretation = "all (including added info)";    break;
      default:
        std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
        break;
      }

    // additional reporting as appropriate
    // YEEK 16 CODE (set by '--yeek')
    if ( xeona::yeek == 16 || xeona::yeek == 1 )
      {
        std::ostringstream put;
        put << "  log rank to GLPK level"                 << "\n"
            << "    logger rank : " << loggerRank         << "\n"
            << "    GLPK level  : " << glpkInterpretation << "\n";
        logger->repx(logga::yeek, "additional reporting follows, yeek", xeona::yeek);
        logger->putx(logga::yeek, put);
      }
    else                                    // normal code
      {
        // completion reporting
        logger->repx(logga::xtra, "using current logger rank"    , loggerRank);
        logger->repx(logga::xtra, "returning GLPK interpretation", glpkInterpretation);
      }

    // return value
    return glpkLevel;
  }
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : logger.cc
//  file-create-date : Thu 05-Apr-2007 11:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : run-time logging functionality / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/logger.cc $
//  LOCAL AND SYSTEM INCLUDES

#include "logger.h"           // companion header for this file (place first)

#include "../c/smart_ptr.h"   // switch easily between TR1 and Boost smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <fstream>            // file-based io (in case of logging to file)
#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <map>                // STL associative container
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <unistd.h>           // sleep(), usleep()

// CAUTION: this unit does not require the -lboost_date_time linking
// directive in the makefile (but some uses of the following header do)

#include <boost/date_time/posix_time/posix_time.hpp>   // plus io
#include <boost/algorithm/string.hpp>        // string recasing, trimming, splitting
#include <boost/algorithm/string_regex.hpp>  // additional regex support
#include <boost/format.hpp>             // printf style formatting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

// PREPROCESSOR MACROS

#if !defined(XE_OUTPUT)
# define XE_OUTPUT 2 // 1 = log to file, 2 = stdlog (std::clog), 3 = stderr (std::cerr)
#endif

// ------------------------------------------------------------
//  notes           : logger concept
// ------------------------------------------------------------
//
//  Free function returning reference to logger object
//
//  Design notes
//
//      The design concept draws on Meyers (1998, pp 219-223)
//      and, in particular, item 47 entitled "Ensure that
//      non-local static objects are initialized before they are
//      used".
//
//      The design objective is to give any number of class
```

```
//      constructors -- some of which are instantiated before the
//      main function -- and the main function itself access to
//      universal logging.  Ideally then, only one Logger object
//      should be created and it should be properly initialized
//      before first use.  Moreover, it is not possible to
//      determine or mandate the order in which calls to such an
//      object might take place.
//
//      The solution is to create a free function in the logga
//      namespace containing a static instance of the Logger
//      class.  This function returns a reference to a
//      necessarily unique Logger instance -- with the first such
//      call giving rise to the correct initialization of the
//      Logger instance.  Cout reporting shows that the Logger
//      constructor and destructor are called only once -- as
//      required.
//
//  CAUTION
//
//      A full (as opposed to forward) class declaration for Logger
//      MUST PRECEDE the free function declaration/definition for
//      refLogStream().
//
//  Usage
//
//      0. Note the typedef within logger.h:
//
//           typdef shared_ptr<Logger> pLogger
//
//      1. Within free and member functions as automatic variable:
//
//         {
//           logga::pLogger logger = logga::ptrLogStream();
//           logger->repx(logga::info, 1, "from function");
//         }
//
//      2. Within classes proper as per object instances:
//
//          class SomeClass
//          {
//          public:
//            SomeClass() :
//              d_logger(logga::ptrLogStream())
//            {
//              d_logger->repx(logga::info, 101, "per object approach");
//            }
//          private:
//            logga::pLogger d_logger;
//          };
//
//      3. Within classes proper as a per class instance:
//
//          class AnotherClass
//          {
//          public:
//            AnotherClass()
//            {
//              s_logger->repx(logga::info, 101, per class approach");
//            }
//          private:
//            static logga:pLogger s_logger;
//          };
//
//          logga::pLogger
//          AnotherClass::s_logger = logga::ptrLogStream();
//
//      See the unit test file for more examples of usage.
//
//  ------------------------------------------------------
//   Additional information regarding compiler macros
//  ------------------------------------------------------
//
//  Design notes
//
//      The member function repx() is hash-define overwritten to
//      additionally report the compiler macros __FILE__, __LINE__,
//      and__func__.
//
//  Control
//
//      This feature can be disabled by modifying the hash-if
```

```
//      directive at the bottom of this file.
//
//  ----------------------------------------------------------
//   Use of function templates and link-time errors
//  ----------------------------------------------------------
//
//  Background
//
//      For a quick synopsis of function templates (aka template
//      functions), see Loudon (2003, pp 101-102).  For a more
//      detailed coverage, consult Lischner (2003, pp 180-184).
//
//  Link-time errors
//
//      It was found that link-time errors would occur when the
//      three-box approach was adopted (using .h and .cc files),
//      for instance:
//
//        "undefined reference to `bool
//         logga::Logger::repx<int>(logga::Rank, int, int)'"
//
//      Oualline (1995, p445) suggests, instead, to place
//      function template definitions in the header file and to
//      also inline them (noting that the inline keyword is a
//      compiler hint only).
//
//      The Oualline approach worked here.  Strangely, C-style
//      strings and string literals were okay with the three-box
//      model, but ints, doubles, and std::strings failed to link.
//
//  ----------------------------------------------------------
//   References
//  ----------------------------------------------------------
//
//      Loudon, Kyle.  2003.  C++ pocket reference.  O'Reilly and
//        Associates, Sebastopol, USA.  ISBN 0-596-00496-6.
//
//      Meyers, Scott.  1998.  Effective C++ : 50 specific ways to
//        improve your programs and design --- Second edition.
//        Addison-Wesley, Boston, USA.  ISBN 0-201-92488-9.
//
//      Oualline, Steve.  1995.  Practical C++ programming -- First
//        edition.  O'Reilly and Associates, Sebastopol, USA.
//        ISBN 1-56592-139-9.
//
//  ----------------------------------------------------------

//  CODE

namespace logga
{
  // ----------------------------------------------------------
  //  VARIABLE        : logga::rankNoData
  //  VARIABLE        : logga::rankJumpy
  // ----------------------------------------------------------

  Rank rankNoData = dbug;
  Rank rankJumpy  = dbug;

  // ----------------------------------------------------------
  //  CLASS           : logga::Logger
  // ----------------------------------------------------------

  //  STATIC DEFINITIONS - with explicit initialization

  bool Logger::s_truncate = true;            // console truncation flag

  // CREATORS

  // ----------------------------------------------------------
  //  MEMBER FUNCTION : Logger::Logger
  // ----------------------------------------------------------

  Logger::Logger
  (std::ostream& os) :
    d_os(os),
    d_was(boost::posix_time::microsec_clock::universal_time()),
    d_lastCall(e_none),
    d_repCount(0),
    d_reportLevel(static_cast<Rank>(xeona::reportLevelOpening)),
    d_enableBeep(false),
```

```
    d_beepCompletion(false),
    d_beepThreshold(logga::yeek),           // and beeping on yeek is disabled
    d_triggerStr(),                         // empty string
    d_triggerRegex(),                       // zero argument construction
    d_finalBlank(false),

    d_returnStatus(-1),                     // nonsensical value
    d_returnInterp(),                       // empty string

    d_yeekNoFirst(0),
    d_killNoFirst(0),
    d_warnNoFirst(0),

    d_yeekAllCount(0),
    d_killAllCount(0),
    d_warnAllCount(0),
    d_infoAllCount(0),
    d_dbugAllCount(0),
    d_xtraAllCount(0),
    d_adhcAllCount(0),

    d_yeekFixedCount(0),
    d_killFixedCount(0),
    d_warnFixedCount(0),
    d_infoFixedCount(0),
    d_dbugFixedCount(0),
    d_xtraFixedCount(0),
    d_adhcFixedCount(0),

    d_yeekResetCount(0),
    d_killResetCount(0),
    d_warnResetCount(0),
    d_infoResetCount(0),
    d_dbugResetCount(0),
    d_xtraResetCount(0),
    d_adhcResetCount(0)
  {
    this->repx(logga::dbug, "constructor call", "using this->repx");
#if 0 // 0 = ignore, 1 = test "delta" timings using a 200ms delay
    this->repx(logga::dbug, "delay set", "200ms");
    usleep(200000);                         // <unistd.h>
#endif // 0
    this->repx(logga::dbug, "timestamp", timestampUTC());
  }

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : Logger::~Logger
  // ------------------------------------------------------------

  Logger::~Logger()
  {
    this->repx(logga::dbug, "destructor call", "using this->repx");

    // print final rule

    if ( d_repCount > 0 )
      {
        printRule(0);                       // zero blank lines to follow
        this->addSmartBlank();
      }

    // display log call print tally if -- at various report
    // levels (noting that 'yeek' is officially silent) -- any
    // log messages were written out OR kill calls were made OR
    // warn calls were made -- note too this logging policy can
    // be readily changed to suit circumstances

    if ( ( d_repCount     > 0 && d_reportLevel >= info )    // logs were printed
         ||
         ( d_killAllCount > 0 && d_reportLevel >= yeek )    // kills were called
         ||
         ( d_warnAllCount > 0 && d_reportLevel >= kill ) )  // warns were called
      {
        std::ostringstream put;
        put << "    Logger destructor call data"     << "\n";
        put << ""                                    << "\n";

        put << summarizeRankCounts();       // normal trailing newline not required
        this->putx(logga::yeek, put);       // always print if here
        this->addSmartBlank();              // set subsequent logging behavior
      }
```

```
  // display application return status, if known

  if ( d_returnStatus >= 0 )               // that is, overwritten construction value
    {
      std::ostringstream put;
      put << "    application exit : " << d_returnStatus
          << " = " << d_returnInterp           << "\n";
      this->putx(logga::yeek, put);        // always print if here
      this->addSmartBlank();               // set subsequent logging behavior
    }

  // this next statement couples with 'std::cout' calls in
  // class 'AppInfo' (the logger stream itself is set in the
  // free function 'logga::ptrLogStream' and need not be
  // 'std::clog') -- the approach is not especially clean but
  // is acceptable, given that '~Logger' is the last hand-coded
  // destructor called

  if ( d_finalBlank )
    {
      d_os << std::flush;
      std::cout << "\n";          // add newline to stdout, even if no logs
      std::cout << std::flush;
    }
  else if ( d_repCount > 0 )
    {
      d_os << "\n";               // add newline to stream, if not above and if logs
      d_os << std::flush;
    }

  if ( d_beepCompletion )
    this->beep(1);                // final beep, comes after 'main' returns

} // ~Logger

// LOGGING FUNCTIONS - most are defined in "logger.h" to protect from the CPP macro

// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::test (overloaded)
// ------------------------------------------------------------

bool
Logger::test                                 // insert a TEST COUNT COMMENCING line
(const int testCount)
{
  d_os << "\n";
  if ( testCount == 0 )                      // special case of final line
    d_os << "   TESTS COMPLETE" << "\n";
  else
    d_os << "   TEST " << testCount << " COMMENCING" << "\n";
  d_os << "\n";
  d_lastCall = e_none;                       // no special treatment from next call
  return true;
}

bool
Logger::test
(const int         testCount,
 const std::string putMsg)                   // 'put'-type message without "\n"
{
  test(testCount);
  d_os << "   ++ " << putMsg << "\n";       // note leading "   ++ "
  d_lastCall = e_addblank;
  return true;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::addSmartBlank (overloaded)
// ------------------------------------------------------------

void
Logger::addSmartBlank()                      // add blank line in a relatively smart way
{
  d_lastCall = e_addblank;                   // simply modify the last call status
}

void
Logger::addSmartBlank                        // ditto, but only if rank-appropriate
(const logga::Rank rank)
{
```

```
  if ( rank > d_reportLevel )              // check verbosity
    {
      return;
    }
  addSmartBlank();                         // action
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::addDumbBlank (overloaded)
// ----------------------------------------------------------

void
Logger::addDumbBlank()
{
  d_os << "\n";                            // call ensures consistent ostream usage
}

void
Logger::addDumbBlank                       // ditto, but only if rank-appropriate
(const logga::Rank rank)
{
  if ( rank > d_reportLevel )              // check verbosity
    {
      return;
    }
  addDumbBlank();                          // action
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::addFinalStdoutBlank
// ----------------------------------------------------------

void
Logger::addFinalStdoutBlank()
{
  d_finalBlank = true;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::flush
// ----------------------------------------------------------

void
Logger::flush()                            // explicitly flush the output buffer
{
  d_os << std::flush;
}

// ACCESSORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::getReportLevelRank
// ----------------------------------------------------------

logga::Rank
Logger::getReportLevelRank() const         // return an enum
{
  return d_reportLevel;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::getReportLevelInt
// ----------------------------------------------------------

unsigned
Logger::getReportLevelInt() const          // return an integer
{
  return static_cast<unsigned>(d_reportLevel);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::getReportLevelStr
// ----------------------------------------------------------

std::string                                // used by 'AppInfo' object
Logger::getReportLevelStr() const          // return an interpretation
{
  std::string interpretation = "(not overwritten)";
  switch( d_reportLevel )
    {
    case yeek: interpretation = "no output";    break;
```

```
      case kill: interpretation = "kills";        break;
      case warn: interpretation = "warnings";     break;
      case info: interpretation = "information";  break;
      case dbug: interpretation = "debug";        break;
      case xtra: interpretation = "extra";        break;
      case adhc: interpretation = "all";          break;
      default: std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
        // CAUTION: do not place 'repx' call here, else 'const' problems
      }
    return interpretation;
  }

  // ------------------------------------------------------
  //  MEMBER FUNCTION : Logger::getAllTriggerStr
  // ------------------------------------------------------

  const std::string
  Logger::getAllTriggerStr() const
  {
    std::string buf;

    buf += formatTriggerStr(yeek, d_yeekAllCount);
    buf += formatTriggerStr(kill, d_killAllCount);
    buf += formatTriggerStr(warn, d_warnAllCount);
    buf += formatTriggerStr(info, d_infoAllCount);
    buf += formatTriggerStr(dbug, d_dbugAllCount);
    buf += formatTriggerStr(xtra, d_xtraAllCount);
    buf += formatTriggerStr(adhc, d_adhcAllCount);

    if ( ! buf.empty() )                    // nonzero length
      buf = buf.substr(1);                  // trim leading char, in this case, a space
    else
      buf = "(no log calls)";

    return buf;
  }

  // ------------------------------------------------------
  //  MEMBER FUNCTION : Logger::summarizeRankCounts
  // ------------------------------------------------------

  const std::string                         // has an uncharacteristic trailing newline
  Logger::summarizeRankCounts() const
  {
    unsigned sumAllCount
      = d_yeekAllCount
      + d_killAllCount
      + d_warnAllCount
      + d_infoAllCount
      + d_dbugAllCount
      + d_xtraAllCount
      + d_adhcAllCount;
    unsigned sumFixedCount
      = d_yeekFixedCount
      + d_killFixedCount
      + d_warnFixedCount
      + d_infoFixedCount
      + d_dbugFixedCount
      + d_xtraFixedCount
      + d_adhcFixedCount;
    unsigned sumResetCount
      = d_yeekResetCount
      + d_killResetCount
      + d_warnResetCount
      + d_infoResetCount
      + d_dbugResetCount
      + d_xtraResetCount
      + d_adhcResetCount;

    std::ostringstream oss;
    oss << "    repx()      call        print           reset" << "\n";
    oss << "    ------------------------------------------" << "\n";
    if ( d_yeekFixedCount > 0 )             // conditional reporting
      {
  formatRankCount(oss, calcRank(yeek), d_yeekAllCount, d_yeekFixedCount, d_yeekResetCount);
      }
  formatRankCount(oss, calcRank(kill), d_killAllCount, d_killFixedCount, d_killResetCount);
  formatRankCount(oss, calcRank(warn), d_warnAllCount, d_warnFixedCount, d_warnResetCount);
  formatRankCount(oss, calcRank(info), d_infoAllCount, d_infoFixedCount, d_infoResetCount);
  formatRankCount(oss, calcRank(dbug), d_dbugAllCount, d_dbugFixedCount, d_dbugResetCount);
  formatRankCount(oss, calcRank(xtra), d_xtraAllCount, d_xtraFixedCount, d_xtraResetCount);
```

```
formatRankCount(oss, calcRank(adhc), d_adhcAllCount, d_adhcFixedCount, d_adhcResetCount);
formatRankCount(oss, "",             sumAllCount,    sumFixedCount,    sumResetCount);

  return oss.str();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::getTrigger
// ----------------------------------------------------------

const std::string
Logger::getTrigger() const
{
  return d_triggerStr;
}

// STATIC MANIPULATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::enableConsoleTruncation
// ----------------------------------------------------------

bool
Logger::enableConsoleTruncation()
{
  s_truncate = true;
  return s_truncate;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::disableConsoleTruncation
// ----------------------------------------------------------

bool
Logger::disableConsoleTruncation()
{
  s_truncate = false;
  return s_truncate;
}

// MANIPULATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::enableBeeping
// ----------------------------------------------------------

bool
Logger::enableBeeping()
{
  d_enableBeep = true;
  return d_enableBeep;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::disableBeeping
// ----------------------------------------------------------

bool
Logger::disableBeeping()
{
  d_enableBeep = false;
  return d_enableBeep;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::setBeepOnCompletion
// ----------------------------------------------------------

void
Logger::setBeepOnCompletion()
{
  d_beepCompletion = true;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::setBeepOnOrAbove
// ----------------------------------------------------------

void
Logger::setBeepOnOrAbove
(logga::Rank rank)
```

```
{
  d_beepThreshold = rank;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::setTrigger
// ----------------------------------------------------------

void
Logger::setTrigger
(const std::string& trigger)
{
  d_triggerStr = trigger;
  boost::regex temp(d_triggerStr);
  d_triggerRegex = temp;

  // CAUTION: place logging statements after the trigger
  // setting process is complete -- otherwise a
  // "/usr/local/include/boost-1_34_1/boost/regex/v4/basic_regex.hpp"
  // assertion may fail on '0 != m_pimpl.get()' and the
  // application is duly "aborted" and core dumped!

  this->repx(logga::dbug, "trigger string set", d_triggerStr);
  this->repx(logga::dbug, "trigger regex set", d_triggerRegex);  // streamable
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::resetRankCounts
// ----------------------------------------------------------

void
Logger::resetRankCounts                    // at given level and noisier
(const unsigned reportLevel)               // default setting given in header
{
  Rank level = static_cast<logga::Rank>(reportLevel);
  switch ( level )
    {
    case yeek: d_yeekResetCount = 0;       // CAUTION: no 'break' statement is correct
    case kill: d_killResetCount = 0;
    case warn: d_warnResetCount = 0;
    case info: d_infoResetCount = 0;
    case dbug: d_dbugResetCount = 0;
    case xtra: d_xtraResetCount = 0;
    case adhc: d_adhcResetCount = 0; break;
    default: std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::setReportLevel
// ----------------------------------------------------------

Rank                                       // return previous report level
Logger::setReportLevel
(const unsigned reportLevel)
{
  // note that out-of-range casting is okay
  Rank newReportLevel = static_cast<Rank>(reportLevel);

  // range check sought level
  if ( newReportLevel < logga::yeek || newReportLevel > logga::adhc )     // not [0,6]
    {
      // may also produce compiler warning: character constant too long for its type
      this->repx(logga::warn, "out-of-range argument, check caller", newReportLevel);
    }

  // capture current (but soon to be previous) value
  Rank oldReportLevel = d_reportLevel;

  // reset and report
  std::ostringstream oss;                  // for reporting
  oss << d_reportLevel << " > "  << newReportLevel;

  if ( newReportLevel > oldReportLevel )   // noisier, 'repx' after resetting
    {
      d_reportLevel = newReportLevel;
      this->repx(logga::dbug, "report level reset from > to", oss.str());
    }
  else                                     // same or quieter, 'repx' before resetting
    {
      this->repx(logga::dbug, "report level reset from > to", oss.str());
```

```
      d_reportLevel = newReportLevel;
    }

  resetRankCounts(newReportLevel);         // CAUTION: note call to reset triggers
  return oldReportLevel;
}

// UTILITY FUNCTIONS

// -----------------------------------------------------------
//  MEMBER FUNCTION : Logger::shouldLog
// -----------------------------------------------------------

bool                                       // 'true' means "should log"
Logger::shouldLog
(const std::string& file)
{
  if ( d_triggerStr.empty() )              // no need to continue
    return false;

  if ( boost::find_regex(file, d_triggerRegex) )
    return true;
  else
    return false;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : Logger::firstCall
// -----------------------------------------------------------

void
Logger::firstCall()
{
  std::cout << std::flush;                 // ensure synchronization
  d_os << "\n";                            // initial opening line
  printHeader();
  printRule(0);                            // zero blank lines to follow
}

// CAUTION: relatively trivial bug: 'printHeader' does not
// respond if the preprocessor macros are "switched off" using
// the 'hash-if' at the end of "logger.h"

// -----------------------------------------------------------
//  MEMBER FUNCTION : Logger::printHeader
// -----------------------------------------------------------

void
Logger::printHeader()
{
  std::ostringstream ssAt;                 // for line, file, and func information
  std::string        strAt;                // string version of above
  int                tab;                  // setw allowance in main stringstream

  ssAt << ""
       << std::setw(s_TAB1 + s_MAR) << std::right << "line"
       << std::setw(         s_GUT) << ""
       << std::setw(s_TAB2 + s_OVR) << std::left  << "source"
       << std::setw(s_TAB3 + s_GUT) << std::left  << "call";

  strAt = ssAt.str();
  tab   = s_TAB1 + s_MAR + s_GUT + s_TAB2 + s_TAB3 + s_GUT;
  boost::trim_right(strAt);                // trim trailing spaces in-situ

  d_os << std::setw(tab)             << std::left  << strAt
       << std::setw(s_TAB4 + s_OVR) << std::right << "no"
       << std::setw(         s_GUT) << ""
       << std::setw(s_TAB5 + s_GUT) << std::left  << "delta-t"
       << std::setw(s_TAB6 + s_GUT) << std::left  << "rank"
       << std::setw(s_TAB7 + s_OVR) << std::left  << "message"
       <<                              std::left  << "comment/value"
       << "\n";

  d_os << std::left;                       // reset to default
  d_repCount = 1;                          // originally initialized to zero
  d_lastCall = e_none;                     // ensure no gap is left
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : Logger::printRule
// -----------------------------------------------------------
```

```
void
Logger::printRule
(const int skip)
{
  // add blank line as needed
  if ( d_lastCall == e_putx )
    d_os << "\n";                          // insert newline
  if ( d_lastCall == e_addblank )
    d_os << "\n";

  // print rule
  d_os << std::string(s_RULE, '.')         // line of dots
       << std::string(skip, '\n')          // series of newlines (usually zero or one)
       << "\n";
  d_lastCall = e_none;                     // ensure no additional gap is left
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::incrementRankCounts
// ------------------------------------------------------------

void
Logger::incrementRankCounts
(unsigned rank)                            // will also accept a logga::Rank
{
  Rank reportLevel = static_cast<Rank>(rank);

  switch ( reportLevel )
    {
    case yeek: ++d_yeekAllCount; break;
    case kill: ++d_killAllCount; break;
    case warn: ++d_warnAllCount; break;
    case info: ++d_infoAllCount; break;
    case dbug: ++d_dbugAllCount; break;
    case xtra: ++d_xtraAllCount; break;
    case adhc: ++d_adhcAllCount; break;
    default: std::clog << "** coding error 03 in source file " << __FILE__ << std::endl;
    }

  // count only printed to console calls
  if ( reportLevel > d_reportLevel ) return;

  switch ( reportLevel )
    {
    case yeek: ++d_yeekFixedCount; ++d_yeekResetCount; break;
    case kill: ++d_killFixedCount; ++d_killResetCount; break;
    case warn: ++d_warnFixedCount; ++d_warnResetCount; break;
    case info: ++d_infoFixedCount; ++d_infoResetCount; break;
    case dbug: ++d_dbugFixedCount; ++d_dbugResetCount; break;
    case xtra: ++d_xtraFixedCount; ++d_xtraResetCount; break;
    case adhc: ++d_adhcFixedCount; ++d_adhcResetCount; break;
    default: std::clog << "** coding error 04 in source file " << __FILE__ << std::endl;
    }

  // update no (number) of first call

  if ( reportLevel == yeek && d_yeekNoFirst == 0 )
    d_yeekNoFirst = d_repCount;
  if ( reportLevel == kill && d_killNoFirst == 0 )
    d_killNoFirst = d_repCount;
  if ( reportLevel == warn && d_warnNoFirst == 0 )
    d_warnNoFirst = d_repCount;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::formatTriggerStr
// ------------------------------------------------------------

std::string
Logger::formatTriggerStr
(logga::Rank rank,
 unsigned    count) const
{
  if ( count == 0 )
    return "";

  // boost::to_lower(term);
  return boost::str(boost::format(" %s (%d)") % calcRank(rank) % count);
}
```

```
// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::formatRankCount
// ------------------------------------------------------------

void
Logger::formatRankCount                    // used by 'summarizeRankCounts'
(std::ostringstream& os,
 const std::string&  term,
 unsigned            call,                 // logger called count
 unsigned            print,                // logger printed count
 unsigned            reset) const          // logger printed count after reset
{
  std::string addn;                        // highlight string for print
  if ( term == calcRank(yeek) && print > 0 )
    addn = boost::str(boost::format("< %d") % d_yeekNoFirst);
  if ( term == calcRank(kill) && print > 0 )
    addn = boost::str(boost::format("< %d") % d_killNoFirst);
  if ( term == calcRank(warn) && print > 0 )
    addn = boost::str(boost::format("< %d") % d_warnNoFirst);
  // boost::to_lower(term);                // downcase alpha characters
  os << "    ";                            // left indent
  os << boost::format("%-4s     ") % term; // print tag
  os << boost::format("%6d       %6d  %-7s %6d") % call % print % addn % reset;
  os << "\n";                              // add trailing newline
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::endOfLogCall
// ------------------------------------------------------------

void
Logger::endOfLogCall                       // only reacts to first kill
(const logga::Rank rank)
  throw(std::exception,                    // exception specification
        xeona::kill_on_log)
{
  // static member
  static bool firstKill = true;            // assignment at initialization only

  // beep behavior
  if ( rank <= d_beepThreshold && rank != logga::yeek )
    this->beep(2);

  // kill behavior
  if ( rank != kill ) return;              // not a kill
  if ( firstKill == false ) return;        // subsequent calls do nothing
  firstKill = false;                       // update status

  // exit on kill if not '--krazy'

  if ( xeona::nopro == true )
    {
      this->repx(logga::info, "skipping xeona::kill_on_log throw", "");
    }
  else
    {
      this->repx(logga::warn, "will throw xeona::kill_on_log", "");

      // CAUTION: this use of throw could be considered an
      // abuse of the exception handling system.  That said, it
      // does shut the application down in a succinct and clean
      // manner.

      throw xeona::kill_on_log();
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : Logger::numToPad
// ------------------------------------------------------------

std::string
Logger::numToPad  // could be better implemented using Boost.Format library
(double input,
 int    LEFT,     // zero-padded digits (default 2) left of decimal point
 int    RIGHT)    // digits (default 4) right of decimal point
{
  //  Purpose     : convert doubles to padded truncated strings
  //  Note        : default values for LEFT and RIGHT provided in logger.h
  //  Used by     : calcInterval()
  //
```

```
  //          0         ->  00.0000
  //          0.0       ->  00.0000
  //          5.003699  ->  05.0037
  //         -0.003699  ->  -0.0037

  std::ostringstream ossBuf;               // formatting buffer
  ossBuf << std::setw(LEFT + RIGHT + 1)    // define overall width, including point
         << std::setfill('0')              // define zero pad character
         << std::fixed                     // adopt fixed formatting
         << std::setprecision(RIGHT)       // define decimal place count
         << input;
  return ossBuf.str();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::calcInterval
// ----------------------------------------------------------

std::string
Logger::calcInterval()
{
  //  Purpose     : return a string in the following formats
  //  Note        : left and right padding can be adjusted here
  //
  //          00.0000s
  //          00.0000m
  //          00.0000h

  std::string interval;                    // return value in one of several formats
  double      input;                       // input for numToPad() function

  // calculate the interval since the last call as a simple Boost time_duration
  // object -- whilst noting that there is no need to use the more complex Boost
  // time_period object that contains end point information thus [start, end)

  boost::posix_time::ptime now;            // current point in time
  boost::posix_time::ptime was;            // point in time of previous call
  boost::posix_time::time_duration dur;    // difference between the above

  now   = boost::posix_time::microsec_clock::universal_time();
  was   = d_was;                           // retrieve value of 'was'
  d_was = now;                             // reset stored 'was' to 'now'
  dur   = now - was;                       // calculate a simple duration

  long hours             = dur.hours();              // normalized hours
  long totalMicroSeconds = dur.total_microseconds(); // total microseconds

  // process string representation of this interval based on interval length

  if ( totalMicroSeconds < 0 )                       // negative, an error
    {
      this->repx(logga::dbug, "negative interval calculated", totalMicroSeconds);
      return "";
    }
  else if ( totalMicroSeconds < 1e6 * 60 )           // sub minute
    {
      input    = static_cast<double>(totalMicroSeconds)/(1e6);
      interval = numToPad(input) + "s";
      return interval;
    }
  else if ( totalMicroSeconds < 1e6 * 60 * 60 )      // sub hour
    {
      input    = static_cast<double>(totalMicroSeconds)/(1e6 * 60);
      interval = numToPad(input) + "m";
      return interval;
    }
  else if ( totalMicroSeconds < 1e6 * 60 * 60 * 100 ) // sub one hundred hours
    {
      input    = static_cast<double>(totalMicroSeconds)/(1e6 * 60 * 60);
      interval = numToPad(input) + "h";
      return interval;
    }
  else                                               // more than one hundred hours
    {
      this->repx(logga::info, "interval exceeds 100 hours", hours);
      return "";
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : Logger::calcRank
```

```
// ------------------------------------------------------

std::string
Logger::calcRank
(const logga::Rank rank) const
{
  switch ( rank )
    {
    case yeek: return "YEEK";              // for use during development only
    case kill: return "KILL";
    case warn: return "WARN";
    case info: return "info";
    case dbug: return "dbug";
    case xtra: return "xtra";
    case adhc: return "adhc";
    default:   return "????";              // should never be here
      std::clog << "** coding error 05 in source file " << __FILE__ << std::endl;
    }
}

// ------------------------------------------------------
//  MEMBER FUNCTION : Logger::timestampUTC
// ------------------------------------------------------
//  Description  : returns truncated timestamp
//  Role         : reporting
//  Techniques   : Boost.Date_time library
//  Status       : complete
//
//  Design notes
//
//      The various Boost time objects can be injected into
//      stringstreams or processed as strings -- with the
//      latter approach selected here.
//
//      The microsec_clock yields 00:15:20.502428 which is then
//      simply truncated to 00:15:20.50
//
//  Boost library resolution and timezone options
//
//        second_clock, microsec_clock
//        local_time, universal_time
//
// ------------------------------------------------------

std::string
Logger::timestampUTC()
{
  boost::posix_time::ptime now;              // current point in time
  boost::posix_time::time_duration time;     // the non-day-date part
  std::string timestamp;                     // return string

  now       = boost::posix_time::microsec_clock::universal_time();
  time      = now.time_of_day();
  timestamp = to_simple_string(time);        // stringify
  timestamp.resize(11);                      // simple truncation to 0.00 seconds
  return timestamp;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : Logger::resetOSS
// ------------------------------------------------------
//  Description  : restore default ostringstream manipulators
//  Role         : anytime manipulators have been hand set
//  Techniques   : low-level calls (unfortunately)
//  Status       : complete and tested for std::boolalpha
//
//  Design notes -- considered but not used
//
//      Note 'basic_ios' member function 'init' -- which I could
//      not get to work (but did not spend much time either).
//
//      Note 'copyfmt(stream)' described in Josuttis (1999 p615)
//      -- which I have not used.
//
//  References
//
//      Josuttis, Nicolai M.  1999.  The C++ Standard Library :
//        a tutorial and reference.  Addison-Wesley, Boston, USA.
//        ISBN // 0-201-37926-0.
//
//      Lischner, Ray.  2003.  C++ in a nutshell : a language and
```

```
//        library reference.  O'Reilly and Associates, Sebastopol,
//        California, USA.  ISBN 0-596-00298-X.
//
// -----------------------------------------------------------

bool
Logger::resetOSS
(std::ostringstream& oss)                   // user-supplied string-stream to be reset
{
  if ( ! oss.good() )                       // defensive programming
    {
      this->repx(logga::warn, "stringstream read state not good", "");
      return false;
    }
  else
    {
      this->repx(logga::xtra, "stringstream read state good", "");
    }

  // Lischner (2003 p205) lists the following io stream defaults:

  oss.exceptions(std::ios::goodbit);
  oss.fill(' ');
  oss.flags(std::ios::skipws|std::ios::dec); // see below
  oss.precision(6);
  oss.width(0);

  // Lischner (2003 p516) describes the set of io stream flags:
  //
  //     alignment      : left, internal, right
  //     base           : dec, hex, oct
  //     float format   : fixed, scientific
  //     other format   : boolalpha, showbase, showpoint, showpos, uppercase
  //     input          : skipws (skip whitespace before input)
  //     flush behavior : unitbuf (flush after each operation)

  return true;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : updateReturnStatus
// -----------------------------------------------------------
//  Description  : gives ~Logger return information for console reporting
//  Role         : invoked near end of 'main' function
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Could considered using an alias but this might be brittle.
//
// -----------------------------------------------------------

void
Logger::updateReturnStatus
(const int         returnStatus,
 const std::string returnInterpetation)    // defaults to ""
{
  this->repx(logga::xtra, "supplied return status", returnStatus);
  d_returnStatus = returnStatus;
  d_returnInterp = returnInterpetation;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : Logger::beep
// -----------------------------------------------------------
//  Description  : beep functionality
//  Role         : anywhere appropriate
//  Techniques   : tries 'beep' from the beep package first
//
//  beep package
//
//      The more exotic beep requires the 'beep (beep the pc
//      speaker any number of ways)' Linux package -- currently
//      tested with version 1.2.2.
//
//      Could also use 'Xlib' and 'XBell' on systems supporting
//      the X Window System (X11).
//
//  Design notes
//
//      This function uses a static 'firstcall' variable to
//      remember its first call state.
```

```
  //
  //  CAUTION: <cstdlib> 'system' call return
  //
  //      See comments in code regarding portability issues.
  //
  // ----------------------------------------------------------

  void
  Logger::beep
  (int type)                                  // magic numbers are not usually recommended
  {
    static bool notLogged = true;             // assignment at initialization only

    int cycle = 0;
    std::string args;

    switch ( type )
      {
      case 1: cycle = 1; args = "-f  400 -l 150"; break;
      case 2: cycle = 1; args = "-f 1100 -l 150"; break;
      default: std::clog << "** coding error 06 in source file " << __FILE__ << std::endl;
      }

    std::string call  = "beep " + args + " 2>/dev/null";
    const char* ccall = call.c_str();

    // flush 'stdout' and 'stdlog' streams to ensure temporal alignment
    this->flush();
    std::cout << std::flush;

    int i = 0;                                // loop counter
    while ( d_enableBeep )
      {
        ++i;
        int ret = system(ccall);              // utility call thru command interpreter
        if ( ret != 0 )                       // CAUTION: guess that system call failed [1]
          {
            // [1] but this is implementation-dependent rather than
            // mandatory C++ behavior which has only been tested using g++

            std::cout << "\007"               // revert to bell character
                      << std::flush;
            if ( notLogged )
              this->repx(logga::info, "system call to beep utility failed", ret);
            notLogged = false;                // update first call state
          }
        if ( i >= cycle ) break;              // halt criteria
        usleep(1000000);                      // sleep 1000ms
      }
    usleep(100000);                           // sleep 200ms
  }

  // ----------------------------------------------------------
  //  FREE FUNCTION   : logga::refLogStream (DEPRECIATED)
  // ----------------------------------------------------------
  //  Category      : free function in xeona namespace
  //  Purpose       : provide a reference to a single logging object
  //  Status        : *depreciated*
  //
  //  Design notes
  //
  //      This free function 'replaces' normal Logger object
  //      construction.
  //
  //      Consult the associated header file for more details.
  //
  // ----------------------------------------------------------

  Logger&
  refLogStream()
  {
    // refLogStream is depreciated so issue a suitable reminder!
    std::clog <<"PLEASE CHANGE FROM refLogStream() TO ptrLogStream()" << "\n";

    // NOTE: activate following code to log to file
    //
    // useful improvements: add timestamp to filename, place file
    // open code in try block

#if (XE_OUTPUT == 1)
```

```
    // file

    const char* filename = "hardcoded.log";  // note problems with 'common.cc' definition
    // 'text' mode is the default, 'out' is writable, 'app' is append
    static std::ofstream logfile(filename, std::ios::out|std::ios::app);
    static Logger s_file(logfile);      // define and initialize a local static object
    s_file.repx(logga::adhc, "binding", filename);
    return s_file;

#elif (XE_OUTPUT == 2)

    // std::clog

//  static Logger s_clog(std::clog);    // define and initialize a local static object
//  s_clog.repx(logga::adhc, "binding", "stdlog");
//  return s_clog;                      // return a reference to it

    static shared_ptr<Logger> s_clog(new Logger(std::clog));
    s_clog->repx(logga::adhc, "binding", "stdlog");
    return *s_clog;                      // return a reference to it

#elif (XE_OUTPUT == 3)

    // not supported

#endif // XE_OUTPUT
  }

  // ------------------------------------------------------------
  //  FREE FUNCTION   : logga::ptrLogStream
  // ------------------------------------------------------------
  //  Category      : free function in xeona namespace
  //  Purpose       : provide a shared pointer to a single logging object
  //  Status        : complete, concept proven
  //
  //  Design notes
  //
  //      This free function 'replaces' normal Logger object
  //      construction.
  //
  //      Consult the associated header file for more details.
  //
  //  Remark
  //
  //      Note the typedef in the header:
  //
  //          typedef shared_ptr<Logger> spLogger
  //
  // ------------------------------------------------------------

  shared_ptr<Logger>
  ptrLogStream()
  {
    // NOTE: to log to file, activate the following code
    //
    // useful improvements: add timestamp to filename, place file
    // open code in try block

#if (XE_OUTPUT == 1)

    // text file

    const char* filename = "hardcoded.log";  // note problems with 'common.cc' definition
    // 'text' mode is the default, 'out' is writable, 'app' is append
    static std::ofstream logfile(filename, std::ios::out|std::ios::app);
    static shared_ptr<Logger> s_file(new Logger(logfile));
    // additional reporting as appropriate
    // YEEK 22 CODE (set by '--yeek')
    if ( xeona::yeek == 22 || xeona::yeek == 1 )
      {
        s_file->repx(logga::adhc, "binding", filename);
      }
    return s_file;                      // return the shared pointer

#elif (XE_OUTPUT == 2)

    // std::clog

    static shared_ptr<Logger> s_log(new Logger(std::clog));
    // additional reporting as appropriate
    // YEEK 22 CODE (set by '--yeek')
```

```
    if ( xeona::yeek == 22 || xeona::yeek == 1 )
      {
        s_log->repx(logga::adhc, "binding", "stdlog");
      }
    return s_log;                       // return the shared pointer

#elif (XE_OUTPUT == 3)

    // std::cerr

    static shared_ptr<Logger> s_log(new Logger(std::cerr));
    // additional reporting as appropriate
    // YEEK 22 CODE (set by '--yeek')
    if ( xeona::yeek == 22 || xeona::yeek == 1 )
      {
        s_log->repx(logga::adhc, "binding", "stderr");
      }
    return s_log;                       // return the shared pointer

#endif // XE_OUTPUT
  }

  // ------------------------------------------------------------
  //  FREE FUNCTION   : logga::checkReportLevel
  // ------------------------------------------------------------

  bool                                      // 'true' if 'trip' would print
  checkReportLevel
  (const logga::Rank trip)
  {
    logga::spLogger logger  = logga::ptrLogStream();
    logga::Rank reportLevel = logger->getReportLevelRank();
    if ( trip > reportLevel ) return false;
    else                      return true;
  }

} // namespace 'logga'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : recorder.cc
//  file-create-date : Mon 26-Apr-2010 12:10 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : summarizing recorder class / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/recorder.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "recorder.h"         // companion header for this file (place first)

#ifndef _XUTEST               // omit for unit testing
# include "../b/domcon.h"     // domain controller entity
# include "../b/gate.h"       // gateway entity
#endif // _XUTEST

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting

//  CODE

#ifdef _XUTEST                // include for unit testing (declarations in header file)

class DomainController { public: std::string getIdAndKind(); };
class Gateway          { public: std::string getIdAndKind(); };

std::string DomainController::getIdAndKind(){ return "domcon-ut-1"; }
std::string Gateway::getIdAndKind(){ return "gate--ut-1"; }

#endif // _XUTEST

// ----------------------------------------------------------
//  CLASS           : EventRecorder::Event (nested class)
// ----------------------------------------------------------

EventRecorder::Event::Event
(const std::string& event,
 const std::string& details,
 const std::string& remark) :
  d_alert(false),
  d_event(event),
  d_details(details),
  d_remark(remark)
```

```
{
}

void
EventRecorder::Event::addAlert
(const std::string& remark)
{
  d_alert = true;
  if ( ! remark.empty() )
    {
      if ( ! d_remark.empty() ) d_remark += " / ";
      d_remark                  += remark;
    }
}

std::string                                 // right trimmed output
EventRecorder::Event::outputEvent
(const unsigned count) const
{
  std::ostringstream oss;
  if ( d_alert ) oss << "  * ";
  else           oss << "    ";
  oss << ""    << std::setw( 3) << std::right << count
      << "   " << std::setw(15) << std::left  << d_event
      << "   " << std::setw(50) << std::left  << d_details
      << "   "                                << d_remark;
  return boost::trim_right_copy(oss.str());  // also trim trailing spaces
}

// ------------------------------------------------------
//  CLASS           : EventRecorder
// ------------------------------------------------------

EventRecorder::EventRecorder
(const std::string title,
 const unsigned    step) :
  d_title(title),
  d_step(step),
  d_events()                                // empty vector
{
  // record the construction
  const std::string event   = "construction";
  const std::string details = "EventRecorder construction";
  const std::string remark  = d_title;
  d_events.push_back(Event(event, details, remark));
}

EventRecorder::~EventRecorder()
{
}

void
EventRecorder::event
(const std::string event,
 const std::string details,
 const std::string remark)
{
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::event
(const std::string event,
 const std::string details,
 const double      value)
{
  const std::string svalue = boost::str(boost::format("%g") % value);
  d_events.push_back(Event(event, details, svalue));
}

void
EventRecorder::event
(const std::string event,
 const std::string details,
 const int         value)
{
  const std::string svalue = boost::str(boost::format("%d") % value);
  d_events.push_back(Event(event, details, svalue));
}

void
```

```
EventRecorder::cta
(const std::string klass,
 const std::string remark)
{
  const std::string event   = "cta";
  d_events.push_back(Event(event, klass, remark));
}

void
EventRecorder::mark
(const std::string            symbol,
 shared_ptr<Gateway>          gate,
 shared_ptr<DomainController> domcon,
 const std::string            remark)
{
  const std::string event = "mark " + symbol;
  const std::string details
    = gate->getIdAndKind()
    + " from " + domcon->getIdAndKind();
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::unmark
(const std::string            symbol,
 shared_ptr<Gateway>          gate,
 shared_ptr<DomainController> domcon,
 const std::string            remark)
{
  const std::string event = "unmark " + symbol;
  const std::string details
    = gate->getIdAndKind()
    + " from " + domcon->getIdAndKind();
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::hop
(shared_ptr<Gateway>          gate,
 shared_ptr<DomainController> fromDomcon,
 shared_ptr<DomainController> toDomcon,
 const std::string            remark)
{
  const std::string event = "hop";
  const std::string details
    = fromDomcon->getIdAndKind()
    + " to "
    + toDomcon->getIdAndKind()
    + " via "
    + gate->getIdAndKind();
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::capset
(shared_ptr<DomainController> domcon,
 shared_ptr<Gateway>          targetGate,
 const std::string            remark)
{
  const std::string event   = "capset";
  const std::string details
    = targetGate->getIdAndKind()
    + " from "
    + domcon->getIdAndKind();
  // get capacitities
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::transolve
(shared_ptr<DomainController> domcon,
 const std::string            remark)
{
  const std::string event   = "transolve";
  const std::string details = domcon->getIdAndKind();
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::note
(const std::string note)
```

```
{
  const std::string event   = "note";
  const std::string details = note;
  const std::string remark  = "";
  d_events.push_back(Event(event, details, remark));
}

void
EventRecorder::alert
(const std::string remark)
{
  if ( d_events.empty() )
    {
      return;
    }
  Event last = d_events.back();
  last.addAlert(remark);
  d_events.pop_back();
  d_events.push_back(last);
}

std::string
EventRecorder::output() const
{
  // CAUTION: currently leaves 'd_events' intact, but could
  // equally well clear the vector

  const std::string header =
    "cnt"
    "   event"
    "             details"
    "                                                 remark";
  std::ostringstream oss;
  oss << "  " << d_title << " summary"   << "\n"
      << "    step = " << d_step         << "\n"
      << "    " << header                << "\n"
      << "    " << std::string(95, '-')  << "\n";

  int count = 0;                               // event counter
  BOOST_FOREACH( Event event, d_events )
    {
      oss << event.outputEvent(count++) << "\n";
    }
  return oss.str();
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xedocs.cc
//  file-create-date : Tue 23-Sep-2008 06:29 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : processing of 'xedoc' entity documentation / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/xedocs.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "xedocs.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <utility>            // STL pair, make_pair()

#include <boost/algorithm/string.hpp>        // string recasing, trimming, splitting
#include <boost/algorithm/string_regex.hpp>  // additional regex support
#include <boost/foreach.hpp>                 // BOOST_FOREACH iteration macro
#include <boost/format.hpp>                  // printf style formatting
#include <boost/regex.hpp>                   // regular expression support

//  CODE

// ----------------------------------------------------------
//  FREE FUNCTION   : ::match
// ----------------------------------------------------------
//  Description  : simple interface for regex/host matching
//  Role         : Xedox searching
//  Techniques   : Boost::String_algo library, unnamed namespace
//  Status       : complete
//
//  Examples
//
//      ::match("ab" , "abc") is 'true'
//      ::match("^ab", "abc") is 'true'
//      ::match("^ab", "zab") is 'false'
//
//      also
//
//      ::match(""   , "abc") is 'false' (typical behavior is true)
//      ::match("ab" , ""   ) is 'false'
//
//  CAUTION: regex library
//
//      This function uses the Boost.String_alg library for regex
//      (regular expression) support and not the Boost.Regex
//      library directly.
```

```
//
// -------------------------------------------------------

namespace
{
  bool                                      // 'true' if 'regex' matches, else 'false'
  match
  (const std::string& regex,                // regex string (eg "^ab")
   const std::string& host)                 // host string  (eg "abc")
  {
    if ( regex.empty() ) return false;      // override "match found" default behavior
    boost::regex pattern(regex);
    return boost::find_regex(host, pattern);
  }

} // unnamed namespace

// -------------------------------------------------------
//  FREE FUNCTION   : ::extractHeader
// -------------------------------------------------------
//  Description  : extract header details from XEDOC entry using a regex
//  Role         : reporting, also acts as a check on XEDOC integrity
//  Techniques   : Boost.Regex library
//  Status       : complete
//
//  Design notes
//
//      The general pattern is: "    header: h/header.h"
//
//      A failure to find this pattern will often indicate a
//      poorly formed XEDOC.
//
//  See also
//
//      Becker (2007 pp347-504) devotes several chapters to the
//      Boost.Regex library.
//
//  CAUTION: script integration
//
//      This code must integrate with the '../scripts/xedoc'
//      script.
//
// -------------------------------------------------------

namespace
{
  bool
  extractHeader
  (const std::string& xedoc,
   std::string&       header)
  {
    // local logging support
    static logga::spLogger logger = logga::ptrLogStream();  // logger bind
    logger->repx(logga::adhc, "entering member function", "");

    // some alternatives regular expressions are offered
    std::string rgxstr;
    rgxstr = "header: ([[:lower:][:digit:]/]+\\.h)";   // thus {a-z     0-9 /} plus ".h"
    rgxstr = "header: ([[:alnum:]/]+\\.h)";            // thus {a-z A-Z 0-9 /} plus ".h"

    // declare dedicated container for search results, 'smatch' is
    // typedef'ed to 'match_results<std::string::const_iterator>'
    boost::smatch match;

    // both the 'regex' constructor and the 'regex_search' function can throw
    try
      {
        boost::regex pattern(rgxstr);        // optional second arg: boost::regex::icase
        if ( boost::regex_search             // partial match variant, else 'regex_match'
             (xedoc,                         // so-called target sequence
              match,                         // search results
              pattern,
              boost::match_default) )        // search options
          {
            header = match[1];               // also known as sub-expression $1
            return true;
          }
        else
          {
            header = "";                     // empty the string
            return false;
```

```
          }
      }
    catch (const boost::regex_error& r )
      {
        logger->repx(logga::warn, "will rethrow exception", "");
        logger->flush();
        std::cout << "** boost::regex_error exception caught and rethrown: "
                  << r.what() << "\n"
                  << std::flush;
        throw;                                // rethrow
      }
    catch ( ... )
      {
        logger->repx(logga::warn, "will rethrow exception", "");
        logger->flush();
        std::cout << "** unspecified exception caught and rethrown" << "\n"
                  << std::flush;
        throw;                                // rethrow
      }
  } // function 'extractHeader'

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS           : Xedocs
// ----------------------------------------------------------

// STATIC DATA

logga::spLogger Xedocs::s_logger = logga::ptrLogStream();   // bind logger on definition

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : Xedocs
// ----------------------------------------------------------

Xedocs::Xedocs() :
  d_fileName(xeona::xedocsFileName),         // set in 'common.cc'
  d_fileContents(xeona::xedocsFileContents), // "loaded" in 'common.cc'
  d_database()
{
  s_logger->repx(logga::dbug, "constructor call", "");
  s_logger->repx(logga::dbug, "xedocs filename", d_fileName);
  s_logger->repx(logga::dbug, "recovered xedocs char count", d_fileContents.length());

#ifdef _XUTEST
  std::ostringstream put;
  put << d_fileContents << "\n";
  s_logger->putx(logga::dbug, put);
#endif // _XUTEST

  if ( makeDatabase() == false )
    {
      s_logger->repx(logga::warn, "makeDatabase returned false", "");
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Xedocs
// ----------------------------------------------------------

Xedocs::~Xedocs()
{
  s_logger->repx(logga::dbug, "destructor call", "");
}

// ACCESSORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : findXedocForClass
// ----------------------------------------------------------
//  Description  : searches for entity class documentation based on class name
//  Role         : direct support for the '--class arg' option
//  Techniques   : returns enum giving find status, provides result by reference
//  Status       : working (but taken out of service for r2402)
// ----------------------------------------------------------

const Xedocs::FindStatus
Xedocs::findXedocForClass
(const std::string soughtClass,
```

```
 std::string&      result)
{
  s_logger->repx(logga::dbug, "entering member function seeking", soughtClass);
  result = "";

  // abandon if database is empty
  if ( d_database.empty() )
    {
      return e_emptyDatabase;
    }

  // hunt using 'std::map::find' member function
  std::map<std::string, std::string>::iterator pos;
  pos = d_database.find(soughtClass);
  if ( pos != d_database.end() )
    {
      s_logger->repx(logga::dbug, "class found", soughtClass);
      result = pos->second;
      return e_classFound;
    }
  else
    {
      s_logger->repx(logga::warn, "class not found", soughtClass);
      return e_classNotFound;
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : findXedocForRegex
// ----------------------------------------------------------
//  Description  : searches for entity class documentation based on regex
//  Role         : direct support for the '--class arg' option
//  Techniques   : returns enum giving find status, provides result by reference
//  Status       : complete, call now placed in function 'main'
// ----------------------------------------------------------

const Xedocs::FindStatus
Xedocs::findXedocForRegex
(const std::string regex,
 std::string&      result)
{
  s_logger->repx(logga::dbug, "entering member function seeking", "\"" + regex + "\"");
  result = "";

  // abandon if database is empty
  if ( d_database.empty() )
    {
      return e_emptyDatabase;
    }

  // loop database
  int count = 0;
  typedef std::pair<std::string, std::string> record;  // CAUTION: for BOOST_FOREACH
  BOOST_FOREACH( record rec, d_database )
    {
      if ( ::match(regex, rec.first) )       // function 'match' defined above
        {
          s_logger->repx(logga::xtra, "regex match found", regex);
          result += rec.second;
          result += "\n\n";                  // add trailing newline plus blank line
          ++count;
          if ( xeona::releaseStatus == false )
            {
              // undertake additional reporting
              std::ostringstream put;
              put << rec.second << "\n";
              s_logger->putx(logga::adhc, put);
            }
        }
    }

  // housekeeping
  if ( count > 0 )
    {
      result.erase(result.length() - 2);     // remove final blank line
      s_logger->repx(logga::info, "regex matches, count", count);
      return e_classFound;
    }
  else
    {
      s_logger->repx(logga::info, "regex not matched", regex);
```

```
      return e_classNotFound;
    }
}

// ------------------------------------------------------
//  MEMBER FUNCTION : dumpClassNames
// ------------------------------------------------------
//  Description  : gives naturally sorted list of map keys
//  Role         : direct support for the '--class *' option
//  Techniques   : simple traverse
//  Status       : working
//
//  Design notes
//
//      Maps maintain sorted keys, see Josuttis (1999 p194).
//
// ------------------------------------------------------

const int                                   // number of classes
Xedocs::dumpClassNames
(std::string& result)                       // newline-separated and sortedlist
{
#if 1 // 0 = omit header information, 1 = add header information

  s_logger->repx(logga::dbug, "entering member function", "with headers");
  result = "";
  std::map<std::string, std::string>::iterator pos;
  for ( pos  = d_database.begin();
        pos != d_database.end();
        ++pos)
    {
      std::string headerName;
      if ( ::extractHeader(pos->second, headerName) == false )
        {
          s_logger->repx(logga::warn, "no header match (check XEDOC)", "");
          headerName = "(not found)";
        }
      else
        {
          s_logger->repx(logga::adhc, "header name", headerName);
        }
      result += boost::str(boost::format("  %-40s %15s") % pos->first % headerName);
      result += "\n";
    }
  return d_database.size();

#else

  s_logger->repx(logga::dbug, "entering member function", "without headers");
  result = "";
  std::map<std::string, std::string>::iterator pos;
  for ( pos  = d_database.begin();
        pos != d_database.end();
        ++pos)
    {
      result += pos->first;
      result += "\n";
    }
  return d_database.size();

#endif // 0
}

// UTILITY FUNCTIONS

// ------------------------------------------------------
//  MEMBER FUNCTION : makeDatabase
// ------------------------------------------------------
//  Description  : fills the xedoc database
//  Role         : indirect support for the '--class arg' option, called by constructor
//  Techniques   : data is collected at compile-time (hence no ancillary file at run-time)
//  Status       : working
//
//  Design notes
//
//      Map insert call based on Josuttis (1999 p203).
//
// ------------------------------------------------------

bool
Xedocs::makeDatabase()
```

```
{
  s_logger->repx(logga::dbug, "entering member function", "");

  std::string buffer(d_fileContents);        // modifiable buffer

  // confirm container is empty
  if ( ! d_database.empty() )
    {
      s_logger->repx(logga::dbug, "database container not empty", "");
      return false;
    }

  // check for information
  if ( buffer.empty() )
    {
      s_logger->repx(logga::dbug, "'xedocs' file empty", "");
      return false;
    }

  // swap record separators for tabs after confirming no tabs
  if ( boost::algorithm::contains(buffer, "\t") )
    {
      s_logger->repx(logga::dbug, "'xedocs' file contains tab chars", "");
      return false;
    }
  boost::algorithm::replace_all(buffer, "\n\n\n", "\t");

  // spit into entity documentation
  std::vector<std::string> xedocs;
  boost::algorithm::split(xedocs,
                          buffer,
                          boost::algorithm::is_any_of("\t"),
                          boost::algorithm::token_compress_on);

  // split into lines
  BOOST_FOREACH( std::string xedoc, xedocs )
    {
      std::vector<std::string> lines;
      boost::algorithm::trim(xedoc);
      boost::algorithm::split(lines,
                              xedoc,
                              boost::algorithm::is_any_of("\n"),
                              boost::algorithm::token_compress_on);

      // split into words
      BOOST_FOREACH( std::string line, lines )
        {
          std::vector<std::string> words;
          boost::algorithm::trim(line);
          boost::algorithm::split(words,
                                  line,
                                  boost::algorithm::is_any_of(" "),
                                  boost::algorithm::token_compress_on);

          // load database as appropriate
          if ( words.size() == 3 )
            {
              if (    words.at(0) == "class"
                   && words.at(1) == ">"
                   && words.at(2).size() > 0 )
                {
                  std::string key   = words.at(2);
                  std::string value = xedoc;
                  if ( d_database.insert(std::make_pair(key, value)).second )
                    {
                      // successful insert
                    }
                  else
                    {
                      // key already exists
                      s_logger->repx(logga::warn, "duplicate class name present", key);
                      return false;
                    }
                }
            }
        }
    }
  s_logger->repx(logga::dbug, "leaving member function, entries", d_database.size());
  return true;
}
```

```
//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xemopt.cc
//  file-create-date : Wed 20-May-2009 16:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : skeleton xem model generator / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/xemopt.cc $
//
//  GENERAL NOTES FOR THIS FILE

#include "xemopt.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting

//  CODE

// STATIC DEFINITIONS

logga::spLogger Xem::s_logger = logga::ptrLogStream();

// ----------------------------------------------------------
//  MEMBER FUNCTION : Xem
// ----------------------------------------------------------

Xem::Xem
(std::ostream& os,                          // can be either console or file ostream
 const int     svnRev,                      // usually 'xeona::svnRev'
 const int     tab) :                       // angle bracket alignment, note default
   d_os(os),
   d_svnRev(svnRev),                        // minimum svn revision
   d_xemgen(tab)                            // passed thru
{
  s_logger->repx(logga::dbug, "constructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Xem
// ----------------------------------------------------------

Xem::~Xem()
{
  s_logger->repx(logga::dbug, "destructor call", "");
}

// ----------------------------------------------------------
```

```
//  MEMBER FUNCTION : head
// -------------------------------------------------------

void
Xem::head()
{
  XemGenerator& x = d_xemgen;                // for convenience

  x.note();
  x.ident("Revision");
  x.ident("Date");
  x.ident("Author");
  x.ident("URL");
  x.meta("role");

  x.rule("program admin");

  // CAUTION: must generate a "" for empty in-strings, hence "\"\""
  // but empty out-strings are okay as simply ""

  x.special("program.last-run");
  x.out("process-id"     , "");
  x.out("run-kind"       , "");
  x.out("used-svn"       , "");
  x.out("simulate-return", "");

  x.special("program.data-format");
  x.in("minimum-svn", boost::str(boost::format("%d") % d_svnRev));

  x.special("program.run-script-settings");
  x.in("script-model-status s"   , "\"incomplete\"");
  x.in("script-run-me b"         , "0");
  x.in("script-option-exittrip i", "1");
  x.in("script-option-nodata b"  , "0");
  x.in("script-option-jumpy b"   , "0");

  x.special("program.r-processing");
  x.in("r-policy i"   , "31");
  x.in("r-title s"    , "\"generated\"");
  x.in("r-plot-list s", "\"\"");
  x.in("r-highlight-output s", "\"\"");

  x.print(d_os);
}

// -------------------------------------------------------
//  MEMBER FUNCTION : mand
// -------------------------------------------------------

void
Xem::mand
(const int          steps,
 const std::string& remark)
{
  XemGenerator& x = d_xemgen;                // for convenience

  x.rule("mandatory entities");

  x.horizon(steps);
  x.overseer();

  if ( ! remark.empty() )
    {
      x.rule("model");
      x.com(remark);
    }

  x.print(d_os);
}

// -------------------------------------------------------
//  MEMBER FUNCTION : rule
// -------------------------------------------------------

void
Xem::rule
(const std::string& annotation)
{
  XemGenerator& x = d_xemgen;                // for convenience

  x.rule(annotation);
```

```
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : more
// ----------------------------------------------------------

void
Xem::more()
{
  XemGenerator& x = d_xemgen;                // for convenience

  x.entity("DomainController", "domain-controller-1");
  x.com("a domain controller entity (the only one provided)");
  x.com("which can take one of a number of commitment modes --");
  x.com("but REQUIRES that the managed entities support the");
  x.com("elected mode");
  x.out("builtin-remark s", "");
  x.in("init-scale-problem b"             , "1");
  x.in("init-use-advanced-initial-basis b", "1");
  x.in("init-use-simplex-presolver b"     , "1");
  x.in("init-use-mip-presolver b"         , "0");
  x.com("these four GLPK solver behavior settings should be set");
  x.com("to true unless run-time tests indicate otherwise");
  x.inq("commitment-mode s", "fin");
  x.com("supported commitment-mode values (lmp is nodal pricing):");
  x.com("fin | ghg | nox | dep | luc | lmp | merit | first");
#if 0 // gateways
  x.inQ("ranked-selgates L", "gate-1");
#else
  x.inq("ranked-selgates L", "");
#endif // 0
  x.com("the ranked-selgates (bridging the right side) must be");
  x.com("null, one, or listed in DESCENDING priority");
#if 0 // asset operators
  x.inQ("asset-operators L", "asop-1");
#else
  x.inq("asset-operators L", "");
#endif // 0
  x.com("the asset-operators may be null, individual, or listed");
  x.com("in no particular order");
#if 0 // demand junctions
  x.inQ("demand-junctions L", "teas-demand-2-split-0");
#else
  x.inq("demand-junctions L", "");
#endif // 0
  x.com("the demand-junctions, which split and join demand, may");
  x.com("be null, individual, or listed in no particular order");
  x.out("variable-costs-financial [$] F"  , "0.0 ..");
  x.out("fixed-costs-financial [$] F"     , "0.0 ..");
  x.hed("b/domcon.h");

  x.print(d_os);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : tail
// ----------------------------------------------------------

void
Xem::tail()
{
  XemGenerator& x = d_xemgen;                // for convenience

  x.rule("tail");

  x.note();
  x.ident("Id");

  x.note();
  x.emacs();

  x.end();
  x.blanks(1);

  x.print(d_os);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : blank
// ----------------------------------------------------------
```

```
void
Xem::blank()
{
  XemGenerator& x = d_xemgen;                // for convenience

  x.blanks(1);

  x.print(d_os);
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : flush
// ------------------------------------------------------------

void
Xem::flush()
{
  d_os << std::flush;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : yeek.cc
//  file-create-date : Tue 17-Nov-2009 11:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : yeek (for running extra code) value interpretation / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/a/yeek.cc $
//
//  GENERAL NOTES FOR THIS FILE
//  LOCAL AND SYSTEM INCLUDES

#include "yeek.h"             // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <iomanip>            // setw() and family
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ---------------------------------------------------------
//  FREE FUNCTION   : xeona::interpret
// ---------------------------------------------------------
//
//  'CASE' preprocessor macro
//
//      'CASE' is simply a programming convenience.  'CASE' is
//      known as a function-like preprocessor macro, see Lischner
//      (2003 p279) for further discussion.
//
//      Hence the statement:
//
//          CASE( 1, "some text");
//
//      maps to:
//
//          case 1: "some text"; break;
//
//      The first argument 'a' (see hash-definition) must be an
//      integer.  The second argument 'b' must be a string
//      literal and may contain matched or unmatched single (')
//      and escaped double quotes (\").
//
//  Example usage
//
//      // comment
//      // YEEK 99 CODE (set by '--yeek')
//      if ( xeona::yeek == 99 )
//        {
```

```
//          s_logger->repx(logga::dbug, "entering yeek code", xeona::yeek);
//          ...
//        }
//
//  CAUTION: retiring of yeek numbers
//
//      Retired yeek numbers should be retired and not recycled.
//      This is because the originating code my be rolled back
//      too.
//
//      In the absence of other considerations, simply comment
//      out the CASE line.
//
//  References
//
//      Lischner, Ray.  2003.  C++ in a nutshell : a language and
//        library reference.  O'Reilly and Associates, Sebastopol,
//        California, USA.  ISBN 0-596-00298-X.
//
// ------------------------------------------------------

#define CASE(a,b) case a: buf = b; break;    // see above comments

namespace xeona
{
  const std::string
  yeekInterpret
  (const unsigned yeekNo)
  {
    std::string buf = "";
    switch (yeekNo)
      {
        // special calls
        CASE(  0, "inactive (provided for convenience)"                                );
        CASE(  1, "provide maximum reporting"                                          );
        // normal calls
        CASE(  2, "show domain-specific OSPs (optimization sub-problems) in web browser");
        CASE(  3, "report from 'OpsFac1Out1_A::downloadSolution' call"                 );
        CASE(  4, "report from 'OptimSubProb::openBnds' calls"                         );
        CASE(  5, "bidset summary from 'AsopLmpBidStatedTs1' constructor"              );
     // CASE(  6, "report from 'OptimSubProb::OptimSubProb' (proper)"                  );
        CASE(  7, "report from 'OptimSubProb::downloadSlack' call"                     );
        CASE(  8, "special test for entity throw code, requires 'AsopBasic' entity"    );
        CASE(  9, "report from 'xeona::geometricProgression' call"                     );
        CASE( 10, "report from 'xeona::isTwoContained' call"                           );
        CASE( 11, "report from 'OpsTransmission_A::downloadSolution' call"             );
        CASE( 12, "report from 'CtlLmpBid_A::uploadBidSet' call"                       );
        CASE( 13, "report from 'TeasHvTransmission::constrain/washup' call"            );
        CASE( 14, "enable block summary report irrespective of --report"               );
        CASE( 15, "write GLPK problem object to file using 'svif::SolverIf::writeInfo'" );
        CASE( 16, "report from 'xeona::logRankToGlpkLevel' call"                       );
        CASE( 17, "report from 'AsopLmpBidAdaptive1' member functions"                 );
        CASE( 18, "report from 'GateStatedTariff::establish' call"                     );
        CASE( 19, "report from 'GateCom<C>::hop' call"                                 );
        CASE( 20, "report from '::registerGateways' call"                              );
        CASE( 21, "report from 'Gateway::recordTransaction' call"                      );
        CASE( 22, "log from 'logga::ptrLogStream' call (normally suppressed)"          );
        CASE( 23, "abandon early after DFS traversal"                                  );
        CASE( 24, "report from 'uploadEngineering' calls"                              );
        CASE( 25, "report from 'svif::SolverIf::resetProblem' call"                    );
        CASE( 26, "report from DFS code = 19 + 27"                                     );
        CASE( 27, "report from 'DomainController::lowestNoTildeSel' call"              );
        CASE( 28, "abandon early during CTA traversal"                                 );
        CASE( 29, "abandon before main loop in 'CtaSimple::captrans' call"             );
        CASE( 30, "report from "
              "'TeasHvTransmission|TeasLoad<C>|TeasSource<C>::constrain' calls"        );
        CASE( 31, "undertake full capacitation in CTA (optional and expensive)"        );
        CASE( 32, "report from 'GateStatedTariff<>::washupSelSide' and "
              "'OfrTariffSet_A::downloadSolution' calls"                               );
        CASE( 33, "add \"(some problem)\" to blank output under "
              "'DataIo::getFieldValue' call and also warn"                             );
        CASE( 34, "report from 'BandedTariffSet::interpretSale' call"                  );
        CASE( 35, "report from 'AsopAdaptiveTs::adapt1' call"                          );

      }
    return buf;
  }
} // namespace 'xeona'

// ------------------------------------------------------
//  FREE FUNCTION   : xeona::yeekSummarize
```

```
// ----------------------------------------------------------

namespace xeona
{
  const std::string                          // contains trailing newline
  yeekSummarize
  (const int indent)
  {
    const int lower =  0;                    // lower attempted yeek number
    const int upper = 99;                    // upper attempted yeek number

    std::ostringstream oss;
    std::string tab(indent, ' ');
    for (int i  = lower;
         i     <= upper;
         ++i)
      {
        const std::string interpretation = yeekInterpret(i);
        if ( interpretation.empty() ) continue;
        oss << tab
            << std::setw(2) << std::setfill(' ') << i
            << " = " << interpretation << "\n";
      }
    return oss.str();
  }
} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : actor.cc
//  file-create-date : Mon 25-Aug-2008 10:35 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : abstract actor entity / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/actor.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "actor.h"            // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// CREATORS

// ------------------------------------------------------------
//  MEMBER FUNCTION : Actor
// ------------------------------------------------------------

Actor::Actor
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  FullEntity(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~Actor
// ------------------------------------------------------------

Actor::~Actor()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : assop.cc
//  file-create-date : Mon 25-Aug-2008 12:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : asset operator entity / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "asop.h"             // companion header for this file (place first)

#include "../b/teas.h"        // technical asset entity
#include "../b/node.h"        // LMP node entity
#include "../b/gate.h"        // gateway entity
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/actor.h"       // actor entity

#include "../c/recset.h"      // records and fields and also record-sets
#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ------------------------------------------------------------
//  MEMBER FUNCTION : AssetOperator
// ------------------------------------------------------------

AssetOperator::AssetOperator
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  CostRegister(record),
  Actor(entityId, record),
  TicToc(commitmentModeSum),
  d_technical_assets(record.tieSingle<std::string>("technical-assets")),
  d_technicalAssets(),                      // empty vector
  d_lmpNodes()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~AssetOperator
// ------------------------------------------------------------
```

```
AssetOperator::~AssetOperator()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// DOMAIN CONTROLLER POINTS OF ENTRY

// ----------------------------------------------------------
//  MEMBER FUNCTION : getTechnicalAssets
// ----------------------------------------------------------

std::vector<shared_ptr<TechnicalAsset> >
AssetOperator::getTechnicalAssets()
{
  s_logger->repx(logga::adhc, "entering member function", "");
  FullEntity::listToVec<TechnicalAsset>(d_technical_assets, d_technicalAssets);
  return d_technicalAssets;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLmpNodes
// ----------------------------------------------------------

std::vector<shared_ptr<LmpNode> >
AssetOperator::getLmpNodes()
{
  s_logger->repx(logga::adhc, "entering member function", "");
  return d_lmpNodes;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : establish
// ----------------------------------------------------------

void
AssetOperator::establish()
{
  // preamble
  s_logger->repx(logga::dbug, "entering member function", "AssetOperator");

  // CAUTION: if required, place own stuff further down the
  // inheritance hierarchy and call this function first:
  // AssetOperator::establish()

  // do own stuff
  CostRegister::resetRegister();             // reset entire register

  // load, report, and step thru technical assets in no required order
  const int technicalAssetCnt
    = FullEntity::listToVec<TechnicalAsset>(d_technical_assets, d_technicalAssets);
  s_logger->repx(logga::adhc, "technical asset count", technicalAssetCnt);
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      ta->CostRegister::resetRegister();     // reset entire register
      ta->establish();
    }

  // step thru LMP nodes in no required order
  // CAUTION: load will take place down the inheritance chain as appropriate
  BOOST_FOREACH( shared_ptr<LmpNode> ln, d_lmpNodes )
    {
      ln->establish();
    }

}

// ----------------------------------------------------------
//  MEMBER FUNCTION : restructure
// ----------------------------------------------------------

void
AssetOperator::restructure
(const xeona::DomainMode commitmentMode)
{
  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      ta->restructure(commitmentMode);
    }
```

```
  // step thru LMP nodes in no required order
  BOOST_FOREACH( shared_ptr<LmpNode> ln, d_lmpNodes )
    {
      ln->restructure(commitmentMode);
    }

  // do own stuff
  TicToc::restructure(commitmentMode);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : initialize
// ----------------------------------------------------------

void
AssetOperator::initialize
(const int                  step,            // CAUTION: the step count is ZERO-based
 shared_ptr<svif::SolverIf> solver)          // solver instance passed thru
{
  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      ta->initialize(step, solver);
    }

  // step thru LMP nodes in no required order
  BOOST_FOREACH( shared_ptr<LmpNode> ln, d_lmpNodes )
    {
      ln->initialize(step, solver);
    }

  // do own stuff
  TicToc::initialize(step, solver);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

void
AssetOperator::washup()
{
  // CAUTION: if required, place own stuff further down the
  // inheritance hierarchy and call this function first:
  // AssetOperator::washup()

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      ta->washup();
    }

  // step thru LMP nodes in no required order
  BOOST_FOREACH( shared_ptr<LmpNode> ln, d_lmpNodes )
    {
      ln->washup();
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : conclude
// ----------------------------------------------------------

void
AssetOperator::conclude()
{
  // CAUTION: if required, place own stuff further down the
  // inheritance hierarchy and call this function first:
  // AssetOperator::conclude()

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      ta->conclude();
    }

  // step thru LMP nodes in no required order
  BOOST_FOREACH( shared_ptr<LmpNode> ln, d_lmpNodes )
    {
      ln->conclude();
    }
```

```
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : setCogenHeatWeight (virtual)
// ----------------------------------------------------------

void
AssetOperator::setCogenHeatWeight
(const double cogenHeatLeadWeight)
{
  s_logger->repx(logga::adhc, "entering member function, weighting", cogenHeatLeadWeight);
  s_logger->repx(logga::warn, "function not overwritten", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getCogenHeatWeight (virtual)
// ----------------------------------------------------------

const double
AssetOperator::getCogenHeatWeight() const
{
  const double invalid = -1.0;
  s_logger->repx(logga::adhc, "entering member function", "");
  s_logger->repx(logga::warn, "function not overwritten", "");
  s_logger->repx(logga::dbug, "about to return invalid value", invalid);
  return invalid;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asop1.cc
//  file-create-date : Wed 15-Apr-2009 21:52 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete asset operators 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop01.cc $
//
//  GENERAL NOTES FOR THIS FILE

#include "asop01.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../b/teas.h"        // technical asset entity
#include "../b/optctl.h"      // control optimization sub-problems for asset operators
#include "../a/exent.h"       // entity exception classes (at writing, just test purposes)

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ----------------------------------------------------------
//  CLASS           : AsopBasic
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopBasic
// ----------------------------------------------------------

AsopBasic::AsopBasic
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId, record, xeona::e_commitmentModes)      // no restriction on mode
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopBasic
// ----------------------------------------------------------

AsopBasic::~AsopBasic()
{
```

```
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopBasic::constrain
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling not required
//  Status       : complete
// ----------------------------------------------------------

const int                                   // number of technical assets processesed
AsopBasic::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // declare some administration counters
  int teasLoop = 0;                         // number of technical assets processed

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoop;
      xeona::putxId(ta, "AsopBasic::constrain technical asset loop");

      // constrain the associated technical asset
      ta->constrain(capacityMode);
    }

  // special test for entity throw code
  // YEEK 8 CODE (set by '--yeek')
  if ( xeona::yeek == 8 )
    {
      s_logger->repx(logga::dbug, "entering yeek code", xeona::yeek);
      s_logger->repx(logga::warn, "this code is strictly for testing", "");
      s_logger->repx(logga::dbug, "will throw xeona::entity_issue", "");
      const std::string msg = "test throw from asset operator '" + getIdAndKind() + "'\n";
      throw xeona::entity_issue(msg);
    }

  // housekeeping
  return teasLoop;
}

// ----------------------------------------------------------
//  CLASS           : AsopPrescribedOrder
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopPrescribedOrder
// ----------------------------------------------------------

AsopPrescribedOrder::AsopPrescribedOrder
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId, record, xeona::e_adminMerit),
  AuxHeatLead(record),
  d_ctl(),                                  // empty shared pointer
  d_ctls()                                  // empty vector
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  d_builtinRemark = "incomplete but well pseudo-coded";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopPrescribedOrder
// ----------------------------------------------------------

AsopPrescribedOrder::~AsopPrescribedOrder()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopPrescribedOrder::constrain
```

```
// -----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling
//  Status       : incomplete but well pseudo-coded
//
//  Design notes
//
//      Merit order uses a technical asset list
//
//          The order in which assets are listed in the technical
//          asset list defines the prescribed order used here.
//          (Whether this order represents "merit" or not is a
//          matter for the modeler!)
//
// -----------------------------------------------------------

const int                                   // number of technical assets processed
AsopPrescribedOrder::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invokation produces a new set of OSPs
  d_ctls.clear();                           // strictly necessary

  // declare some administration counters
  int teasLoop = 0;                         // number of technical assets processed

  // declare a rank counter
  int rank     = 0;                         // rank sequence { 1, 2, 3, 4, .. }

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoop;
      xeona::putxId(ta, "AsopPrescribedOrder::constrain technical asset loop");

      // create and fill a label object
      Label lab(asopId);

      // set the heat lead weighting (with 0.0 = power led and 1.0 = heat led)
      const double heatLeadtWeight = getCogenHeatLeadWeight();
      ta->setCogenHeatWeight(heatLeadtWeight);

      // constrain the associated technical asset (also set floor and ceiling duties)
      const int opsDutyGol = ta->constrain(capacityMode);

      // check for meaningful coupling or loop again
      if ( opsDutyGol == 0 ) continue;

      // recreate and label new control OSP of the required type
      d_ctl.reset(new CtlMeritOrder(d_solver, d_commitmentMode));
      d_ctl->loadOspLabel(lab.str());

      // obtain some technical asset information
      const double upperCapacity = ta->getCeilingDuty();

      // OSP upload call
      int ctlDutyGol;
      boost::tie(ctlDutyGol) = d_ctl->uploadRank(++rank,          // prescribed order
                                                 upperCapacity);  // "big M"-style code

      // OSP couple call (from unit 'b/optprob')
      xeona::couple(d_solver,
                    opsDutyGol,
                    ctlDutyGol,
                    lab.str("couple"));

      // store some information
      d_ctls.push_back(d_ctl);

    } // technical assets loop

  // housekeeping
  return teasLoop;
```

```
}

// ----------------------------------------------------------
//  CLASS           : AsopInternalCosts
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopInternalCosts
// ----------------------------------------------------------

AsopInternalCosts::AsopInternalCosts
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId, record, xeona::e_shortrunModes),
  CostRegisterAsop(record),
  d_ctl(),                                  // empty shared pointer
  d_ctls()                                  // empty vector
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopInternalCosts
// ----------------------------------------------------------

AsopInternalCosts::~AsopInternalCosts()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopInternalCosts::constrain
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling not required, own costs added
//  Status       : first-pass complete
//
//  Design notes
//
//      This asset operator does not need to overwrite the cost
//      minimization objective supplied by the technical assets
//      -- hence the lack of duty coupling code.
//
// ----------------------------------------------------------

const int                                   // number of technical assets processesed
AsopInternalCosts::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invokation produces a new set of OSPs
  d_ctls.clear();                           // strictly not necessary in this case

  // declare some loop counters
  int teasLoop = 0;                         // number of technical assets processed

  // create a label object
  Label lab(asopId);
  lab << "constrain";

  // recreate and label new control OSP of the required type
  d_ctl.reset(new CtlLeastCost(d_solver, d_commitmentMode));
  d_ctl->loadOspLabel(lab.str());

  // upload specific costs -- in this case, the standing costs
  d_ctl->uploadShortrunCosts(d_standingCosts);    // loaded as incremental "shift" costs

  // store some information
  d_ctls.push_back(d_ctl);

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
```

```
      // keep count
      ++teasLoop;
      xeona::putxId(ta, "AsopInternalCosts::constrain technical asset loop");

      // constrain the associated technical asset
      ta->constrain(capacityMode);

    } // technical assets loop

  // housekeeping
  return teasLoop;

} // member function 'constrain'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asop02.cc
//  file-create-date : Thu 09-Jul-2009 15:44 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete asset operators 2 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop02.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "asop02.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../b/teas.h"        // technical asset entity
#include "../b/optctl.h"      // control optimization sub-problems for asset operators
#include "../b/node.h"        // LMP node entity
#include "../b/lmpbid.h"      // LMP auction bidset
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../a/exent.h"       // entity exception classes

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

//  CODE

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::obtainBidsetDialog
// ------------------------------------------------------------

namespace xeona
{
  std::string
  obtainBidsetDialog
  (const double       capacity,             // current capacity
   const std::string& intro)                // optional intro text with trailing newline
  {
    // initial reporting
    logga::spLogger logger = logga::ptrLogStream();
    logger->repx(logga::dbug, "entering free function, capacity", capacity);
    logger->addDumbBlank();
```

```
    // user-defined constants
    const std::string rule  = "-----\n";     // used to demark this function
    const double scaleBand  = 1.0e+06;       // band scale factor
    const double scalePrice = 1.0e-09;       // price scale factor

    // bid delimiter
    const std::string bd    = xeona::modelBidDelim;

    // declare the read buffers
    std::string bufBand;                     // capacity band buffer
    std::string bufPrice;                    // unit price buffer

    // scaled capacity as string
    std::ostringstream tmp;
    tmp << capacity / scaleBand;
    const std::string cap = tmp.str();

    // create standard messages
    std::ostringstream msg;
    msg << "create a bidset by entering multiple space-separated band/price pairs"
        << ", finish early with Ctrl-D if required"                 << "\n"
        << "order not significant, scientific notation supported"
        << ", \"cap\" as alias for current capacity : " << capacity << "\n"
        << "\n";
    const std::string msgstr = msg.str();
    std::ostringstream ask;
    ask << "enter band (" << scaleBand << ") and price (" << scalePrice << ")";
    const std::string askstr = ask.str();

    // terminal output starts
    std::cerr << rule;
    if ( ! intro.empty() ) std::cerr << intro << "\n";
    std::cerr << msgstr;
    std::cerr << std::flush;

    // bidset string-stream
    std::ostringstream bidset;
    // bid << std::fixed;
    // bid << std::showpoint;

    // main loop
    double totalBand = 0.0;                  // running total
    while ( true )
      {
        // input request
        std::cerr << "  " << askstr << " : ";
        std::cin  >> bufBand >> bufPrice;    // two buffers

        // abandon on ctrl-d
        if ( ! std::cin )                    // meaning ctrl-d received
          {
            std::cerr << "complete" << "\n"; // newline also required
            std::cerr << "\n";
            std::cerr << std::flush;
            break;
          }

        // capacity swap on "cap"
        if ( bufBand == "cap" )              // meaning "cap" entered
          {
            bufBand = cap;
          }

        // cast and reject if necessary
        double band  = -1.0;
        double price = -1.0;
        try
          {
            band  = boost::lexical_cast<double>(bufBand);
            price = boost::lexical_cast<double>(bufPrice);
          }
        catch( const boost::bad_lexical_cast& e )
          {
            std::cerr << "  invalid entry rejected" << "\n";
            std::cerr << std::flush;
            continue;
          }

        // check if negative and reject
        if ( band < 0.0 || price < 0.0 )
          {
```

```
            std::cerr << "  negative entry rejected" << "\n";
            std::cerr << std::flush;
            continue;
          }

        // rescale
        band  *= scaleBand;
        price *= scalePrice;

        // incrementally construct the bidset
        if ( ! bidset.str().empty() )        // meaning prior bids exist
          {
            bidset << " " << bd << " ";
          }
        bidset << band << " " << price;

        // break on sufficient capacity
        totalBand += band;
        if ( totalBand >= capacity )
          {
            std::cerr << "\n" << "break on sufficient capacity : " << totalBand << "\n";
            std::cerr << std::flush;
            break;
          }

      } // while true block

    // rework an empty bidset
    std::string bidsetstr = bidset.str();
    if ( bidsetstr.empty() )
      {
        bidsetstr = "0.0 0.0";
      }

    //  terminal output completes
    std::cerr << "capacity ratio : " << totalBand / capacity << "\n"; // div-zero is 'nan'
    std::cerr << "bidset string  : " << "\"" << bidsetstr << "\"" << "\n";
    std::cerr << rule;
    std::cerr << std::flush;

    logger->addSmartBlank();

    return bidsetstr;
  }

} // namespace 'xeona'

// ----------------------------------------------------------
//  CLASS           : AsopGrid
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopGrid
// ----------------------------------------------------------

AsopGrid::AsopGrid
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId, record, xeona::e_auctionLmp),
  d_lmp_nodes(record.tieSingle<std::string>("lmp-nodes"))
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopGrid
// ----------------------------------------------------------

AsopGrid::~AsopGrid()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// DOMAIN CONTROLLER POINTS OF ENTRY

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLmpNodes
// ----------------------------------------------------------
```

```
std::vector<shared_ptr<LmpNode> >
AsopGrid::getLmpNodes()
{
  s_logger->repx(logga::adhc, "entering member function", "");
  FullEntity::listToVec<LmpNode>(d_lmp_nodes, d_lmpNodes);
  return d_lmpNodes;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : establish
// ----------------------------------------------------------

void
AsopGrid::establish()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "AsopGrid");

  // call the base function for the underlying work
  AssetOperator::establish();

  // load LMP nodes and report (will shortly be undertaken for
  // technical assets by class 'AssetOperator')
  const int lmpNodeCnt = FullEntity::listToVec<LmpNode>(d_lmp_nodes, d_lmpNodes);
  s_logger->repx(logga::adhc, "lmp node count", lmpNodeCnt);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling
//  Status       : complete but not tested
// ----------------------------------------------------------

const int                          // number of technical assets and LMP nodes processed
AsopGrid::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "AsopGrid");

  // preamble
  const std::string asopId = getIdentifier();

  // declare some administration counters
  int teasLoops = 0;                       // number of technical assets processed
  int nodeLoops = 0;                       // number of LMP nodes processed

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoops;
      xeona::putxId(ta, "AsopGrid::constrain technical asset loop");

      // constrain the associated technical asset
      const int ret = ta->constrain(capacityMode);

      // CAUTION: transmission assets are not bid coupled and
      // hence return zero in order to "stay out of the loop" --
      // complain if meaningful coupling is indicated
      if ( ret != 0 )
        {
          s_logger->repx(logga::warn, "meaningful coupling encountered", ret);
        }

    } // technical assets loop

  // step thru LMP nodes in no required order
  BOOST_FOREACH( shared_ptr<LmpNode> ln, d_lmpNodes )
    {
      // keep count
      ++nodeLoops;
      xeona::putxId(ln, "AsopGrid::constrain LMP node loop");

      // constrain the associated technical asset
      const int ret = ln->constrain(capacityMode);

      // CAUTION: LMP nodes are not bid coupled and hence return
```

```
      // zero in order to "stay out of the loop" -- complain if
      // meaningful coupling is indicated
      if ( ret != 0 )
        {
          s_logger->repx(logga::warn, "meaningful coupling encountered", ret);
        }

    } // LMP nodes loop

  // return combined count
  return teasLoops + nodeLoops;

} // function 'AsopGrid::constrain'

// ------------------------------------------------------------
//  CLASS           : AsopLmpBidStatedTs1
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : AsopLmpBidStatedTs1
// ------------------------------------------------------------

AsopLmpBidStatedTs1::AsopLmpBidStatedTs1
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId,
                record,
                xeona::e_auctionLmp),        // commitment mode sum
  CostRegisterAsop(record),
  d_market_side(record.tieSingle<std::string>("market-side")),
  d_bidsets_1(record.tieTimeseries<std::string>("bidsets-1")),
  d_marketSide(xeona::e_notSpecified),
  d_bidsets1(),
  d_ctl(),
  d_ctls()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";

  // record the market side or else complain about given 'd_market_side' value
  if      ( d_market_side == "demand" ) d_marketSide = xeona::e_demandSide;
  else if ( d_market_side == "supply" ) d_marketSide = xeona::e_supplySide;
  if ( d_marketSide == xeona::e_notSpecified )
    {
      s_logger->repx(logga::warn, "market side string not valid", d_market_side);
    }

  // load bidset vector at outset
  BOOST_FOREACH( std::string s, *d_bidsets_1 )
    {
      shared_ptr<LmpBidSet> bidset(new LmpBidSet("AsopLmpBidStatedTs1"));
      if ( bidset->pushString(s) == 0 )
        {
          s_logger->repx(logga::warn, "bidset string parse issue, input", s);
        }
      if ( d_marketSide == xeona::e_demandSide ) bidset->negate(); // [1]
      d_bidsets1.push_back(bidset);

      // [1] demand-side curves naturally slope down, therefore
      // the bidset unit prices -- which represent the slope
      // piecewise -- need to multiplied by minus one

      // additional reporting as appropriate
      // YEEK 5 CODE (set by '--yeek')
      if ( xeona::yeek == 5 || xeona::yeek == 1 )
        {
          std::ostringstream put;
          shared_ptr<LmpBidSet> temp = d_bidsets1.back();
          put << "  bidset, market side: " << d_market_side << "\n";
          put << temp->summarizeAll(2);
          s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
          s_logger->putx(logga::dbug, put);
        }
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~AsopLmpBidStatedTs1
// ------------------------------------------------------------
```

```
AsopLmpBidStatedTs1::~AsopLmpBidStatedTs1()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling
//  Status       : first-pass complete
// ----------------------------------------------------------

const int                                   // number of technical assets processed
AsopLmpBidStatedTs1::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "AsopLmpBidStatedTs1");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invocation produces a new set of OSPs
  d_ctls.clear();                           // strictly necessary

  // declare some administration counters
  int teasLoops = 0;                        // number of technical assets processed

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoops;
      xeona::putxId(ta, "AsopLmpBidStatedTs1::constrain technical asset loop");

      // useful messaging data
      const std::string aoid = getIdAndKind();
      const std::string taid = ta->getIdAndKind();

      // create and fill a label object
      Label lab(asopId);

      // constrain the associated technical asset (also set floor and ceiling duties)
      const int opsDutyGol = ta->constrain(capacityMode);

      // check for non-meaningful coupling (transmissions assets)
      // then complain and loop again
      if ( opsDutyGol == 0 )
        {
          s_logger->repx(logga::warn, "uncoupled asset encountered", taid);
          std::ostringstream put;
          put << "  LMP asset operator encountered technical asset that"
              << " does not require duty coupling"                          << "\n"
              << "    asset operator class  : " << "AsopLmpBidStatedTs1"   << "\n"
              << "    asset operator id     : " << aoid                     << "\n"
              << "    technical operator id : " << taid                     << "\n";
          s_logger->putx(logga::xtra, put);
          s_logger->repx(logga::adhc, "will abandon current loop, count", teasLoops);
          continue;
        }

      // recreate and label new control OSP of the required type
      d_ctl.reset(new CtlLmpBid(d_solver, d_commitmentMode));
      d_ctl->loadOspLabel(lab.str("lmp-stated-ts1"));

      // grab current bidset
      shared_ptr<LmpBidSet> bidset = d_bidsets1.at(d_step);

      // test that bidset
      if ( ! bidset )                       // empty bidset
        {
          s_logger->repx(logga::warn, "empty bidset encountered, loop", teasLoops);
          std::ostringstream put;
          put << "  unexpected empty bidset"                      << "\n"
              << "    bidset pointer     : " << bidset            << "\n"
              << "    step               : " << d_step            << "\n"
              << "    asset operator  id : " << aoid              << "\n"
              << "    technical asset id : " << taid              << "\n";
          s_logger->putx(logga::dbug, put);
```

```
          s_logger->repx(logga::adhc, "will abandon current loop, count", teasLoops);
          continue;
        }

      // obtain technical asset information
      const double ceilingDuty     = ta->getCeilingDuty();  // from 'TechnicalAsset'

      // check for asset capacitation through insufficient bidding
      const double hiQuantity = bidset->getHiQuantity();
      if ( ceilingDuty > hiQuantity )
        {
          std::ostringstream oss;
          oss << ceilingDuty << " : " << hiQuantity;
          s_logger->repx(logga::rankJumpy, "asset capacitation, asset : bid", oss.str());
        }

      // OSP upload bidset call
      int ctlDutyGol;
      boost::tie(ctlDutyGol) = d_ctl->uploadBidSet(bidset);

      // OSP couple call (from unit 'b/optprob')
      xeona::couple(d_solver,
                    opsDutyGol,
                    ctlDutyGol,
                    lab.str("couple"));

      // store some information
      d_ctls.push_back(d_ctl);

    } // technical assets loop

  // return
  return teasLoops;

} // function 'AsopLmpBidStatedTs1::constrain'

// ------------------------------------------------------------
//  CLASS           : AsopLmpBidDialog
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : AsopLmpBidDialog
// ------------------------------------------------------------

AsopLmpBidDialog::AsopLmpBidDialog
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId,
                record,
                xeona::e_auctionLmp),        // commitment mode sum
  CostRegisterAsop(record),
  d_market_side(record.tieSingle<std::string>("market-side")),
  d_marketSide(xeona::e_notSpecified),
  // d_bidsets1(),
  d_ctl(),
  d_ctls()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";

  // record the market side or else complain about given 'd_market_side' value
  if      ( d_market_side == "demand" ) d_marketSide = xeona::e_demandSide;
  else if ( d_market_side == "supply" ) d_marketSide = xeona::e_supplySide;
  if ( d_marketSide == xeona::e_notSpecified )
    {
      s_logger->repx(logga::warn, "market side string not valid", d_market_side);
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~AsopLmpBidDialog
// ------------------------------------------------------------

AsopLmpBidDialog::~AsopLmpBidDialog()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : constrain
```

```
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling, 'xeona::obtainBidsetDialog'
//  Status       : first-pass complete
// ----------------------------------------------------------

const int                                   // number of technical assets processed
AsopLmpBidDialog::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "AsopLmpBidDialog");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invocation produces a new set of OSPs
  d_ctls.clear();                           // strictly necessary

  // declare some administration counters
  int teasLoops = 0;                        // number of technical assets processed

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoops;
      xeona::putxId(ta, "AsopLmpBidDialog::constrain technical asset loop");

      // useful messaging data
      const std::string aoid = getIdAndKind();
      const std::string taid = ta->getIdAndKind();

      // create and fill a label object
      Label lab(asopId);

      // constrain the associated technical asset (also set floor and ceiling duties)
      const int opsDutyGol = ta->constrain(capacityMode);

      // check for non-meaningful coupling (transmissions assets)
      // then complain and loop again
      if ( opsDutyGol == 0 )
        {
          s_logger->repx(logga::warn, "uncoupled asset encountered", taid);
          std::ostringstream put;
          put << "  LMP asset operator encountered technical asset that"
              << " does not require duty coupling"                          << "\n"
              << "    asset operator class  : " << "AsopLmpBidDialog"      << "\n"
              << "    asset operator id     : " << aoid                     << "\n"
              << "    technical operator id : " << taid                     << "\n";
          s_logger->putx(logga::xtra, put);
          s_logger->repx(logga::adhc, "will abandon current loop, count", teasLoops);
          continue;
        }

      // recreate and label new control OSP of the required type
      d_ctl.reset(new CtlLmpBid(d_solver, d_commitmentMode));
      d_ctl->loadOspLabel(lab.str("lmp-stated-ts1"));

      // create intro string for bidset dialog
      std::ostringstream intro;
      intro << "call from"
            << " AsopLmpBidDialog operator " << "'" << aoid << "'"
            << " concerning asset "          << "'" << taid << "'"
            << "\n";

      // grab bid
      const double size           = ta->getCeilingDuty();   // current value
      const std::string bidsetstr = xeona::obtainBidsetDialog(size, intro.str());

      const std::string bidsetLabel = "AsopLmpBidDialog interactive";      // improve
      shared_ptr<LmpBidSet> bidset(new LmpBidSet(bidsetLabel));
      bidset->pushString(bidsetstr);

      // test that bidset
      if ( ! bidset )                       // empty bidset
        {
          s_logger->repx(logga::warn, "empty bidset encountered, loop", teasLoops);
          std::ostringstream put;
          put << "  unexpected empty bidset"                      << "\n"
```

```
              << "    bidset pointer     : " << bidset            << "\n"
              << "    step               : " << d_step            << "\n"
              << "    asset operator  id : " << aoid              << "\n"
              << "    technical asset id : " << taid              << "\n";
            s_logger->putx(logga::dbug, put);
            s_logger->repx(logga::adhc, "will abandon current loop, count", teasLoops);
            continue;
          }

        // obtain technical asset information
        const double ceilingDuty     = ta->getCeilingDuty();  // from 'TechnicalAsset'

        // check for asset capacitation through insufficient bidding
        const double hiQuantity = bidset->getHiQuantity();
        if ( ceilingDuty > hiQuantity )
          {
            std::ostringstream oss;
            oss << ceilingDuty << " : " << hiQuantity;
            s_logger->repx(logga::rankJumpy, "asset capacitation, asset : bid", oss.str());
          }

        // OSP upload bidset call
        int ctlDutyGol;
        boost::tie(ctlDutyGol) = d_ctl->uploadBidSet(bidset);

        // OSP couple call (from unit 'b/optprob')
        xeona::couple(d_solver,
                      opsDutyGol,
                      ctlDutyGol,
                      lab.str("couple"));

        // store some information
        d_ctls.push_back(d_ctl);

      } // technical assets loop

    // return
    return teasLoops;

} // function 'AsopLmpBidDialog::constrain'

// ------------------------------------------------------------
//  CLASS           : AsopLmpBidAdaptive1
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : AsopLmpBidAdaptive1
// ------------------------------------------------------------

AsopLmpBidAdaptive1::AsopLmpBidAdaptive1
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId,
                record,
                xeona::e_auctionLmp),        // commitment mode sum
  CostRegisterAsop(record),
  d_market_side(record.tieSingle<std::string>("market-side")),
  d_openingBidset(record.tieSingle<std::string>("opening-bidset")),
  d_targetCommitment(record.tieSingle<double>("target-commitment")),
  d_relativeHysteresis(record.tieSingle<double>("relative-hysteresis")),
  d_priceFactor(record.tieSingle<double>("price-factor")),
  d_priceDelta(record.tieSingle<double>("price-delta")),
  d_submittedBidsets(record.tieTimeseries<std::string>("submitted-bidsets")),
  d_marketSide(xeona::e_notSpecified),
  d_bidset(),
  d_bidsets(),
  d_ctl(),
  d_ctls()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";

  // record the market side or else complain about given 'd_market_side' value
  if      ( d_market_side == "demand" ) d_marketSide = xeona::e_demandSide;
  else if ( d_market_side == "supply" ) d_marketSide = xeona::e_supplySide;
  if ( d_marketSide == xeona::e_notSpecified )
    {
      s_logger->repx(logga::warn, "market side string not valid", d_market_side);
    }
```

```
  // complain if more that one technical asset present
  if ( d_technicalAssets.size() > 1 )
    {
      s_logger->repx(logga::warn,
                     "only one technical asset supported",
                     d_technicalAssets.size());
    }

  // range check adaptive parameters
  if ( d_relativeHysteresis < 0.0 )
    {
      s_logger->repx(logga::rankJumpy,
                     "negative relative hysteresis",
                     d_relativeHysteresis);
    }
  if ( d_priceFactor < 0.0 )
    {
      s_logger->repx(logga::rankJumpy,
                     "negative price factor",
                     d_priceFactor);
    }
  if ( d_priceDelta < 0.0 )
    {
      s_logger->repx(logga::rankJumpy,
                     "negative price delta",
                     d_priceDelta);
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopLmpBidAdaptive1
// ----------------------------------------------------------

AsopLmpBidAdaptive1::~AsopLmpBidAdaptive1()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : establish
// ----------------------------------------------------------

void
AsopLmpBidAdaptive1::establish()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "");

  // call the base function for the underlying work
  AssetOperator::establish();

  // create opening bidset
  const std::string label = "common-bidset";
  d_bidset.reset(new LmpBidSet(label));
  d_bidset->pushString(d_openingBidset);

  // additional reporting as appropriate
  // YEEK 17 CODE (set by '--yeek')
  if ( xeona::yeek == 17 || xeona::yeek == 1 )
    {
      const std::string bidsetstr = d_bidset->stringify();
      std::ostringstream put;
      put << "  yeek 17 reporting"                                    << "\n"
          << "    function               : AsopLmpBidAdaptive1::" << __func__ << "\n"
          << "    bidset label           : " << label                << "\n"
          << "    opening-bidset         : " << d_openingBidset      << "\n"
          << "    opening bidset string  : " << bidsetstr            << "\n"
          << "    target commitment      : " << d_targetCommitment   << "\n"
          << "    relative hysteresis    : " << d_relativeHysteresis << "\n"
          << "    price factor           : " << d_priceFactor        << "\n"
          << "    price delta            : " << d_priceDelta         << "\n";
      s_logger->repx(logga::yeek, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::yeek, put);
      s_logger->addSmartBlank(logga::yeek);
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : constrain call
```

```
//  Role         : standard call
//  Techniques   : duty coupling
//  Status       : first-pass complete but buggy
//
//  Design notes
//
//      At present only one 'd_bidset' of type
//      'shared_ptr<LmpBidSet>' exists.  If more than one
//      technical asset is to be supported then a vector of
//      bidsets is needed.
//
// -----------------------------------------------------------

const int                                   // number of technical assets processed
AsopLmpBidAdaptive1::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "AsopLmpBidAdaptive1");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invocation produces a new set of OSPs
  d_ctls.clear();                           // strictly necessary

  // declare some administration counters
  int teasLoops = 0;                        // number of technical assets processed

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoops;
      xeona::putxId(ta, "AsopLmpBidAdaptive1::constrain technical asset loop");

      // useful messaging data
      const std::string aoid = getIdAndKind();
      const std::string taid = ta->getIdAndKind();

      // create and fill a label object
      Label lab(asopId);

      // constrain the associated technical asset (also set floor and ceiling duties)
      const int opsDutyGol = ta->constrain(capacityMode);

      // check for non-meaningful coupling (transmissions assets)
      // then complain and loop again
      if ( opsDutyGol == 0 )
        {
          s_logger->repx(logga::warn, "uncoupled asset encountered", taid);
          std::ostringstream put;
          put << "  LMP asset operator encountered technical asset that"
              << " does not require duty coupling"                    << "\n"
              << "    asset operator class  : " << "AsopLmpBidAdaptive1"   << "\n"
              << "    asset operator id     : " << aoid                   << "\n"
              << "    technical operator id : " << taid                   << "\n";
          s_logger->putx(logga::xtra, put);
          s_logger->repx(logga::adhc, "will abandon current loop, count", teasLoops);
          continue;
        }

      // recreate and label new control OSP of the required type
      d_ctl.reset(new CtlLmpBid(d_solver, d_commitmentMode));
      d_ctl->loadOspLabel(lab.str("lmp-adaptive-1"));

      // test current bidset
      if ( ! d_bidset )                     // empty bidset
        {
          s_logger->repx(logga::warn, "empty unadapted bidset encountered", "");
          std::ostringstream put;
          put << "  unexpected empty bidset"                    << "\n"
              << "    comment            : " << "unadapted"     << "\n"
              << "    bidset pointer     : " << d_bidset        << "\n"
              << "    step               : " << d_step          << "\n"
              << "    asset operator  id : " << aoid            << "\n"
              << "    technical asset id : " << taid            << "\n";
          s_logger->putx(logga::dbug, put);
          s_logger->repx(logga::adhc, "will abandon current loop, count", teasLoops);
          continue;
        }
```

```
// obtain technical asset information
const double ceilingDuty = ta->getCeilingDuty();     // from 'TechnicalAsset'
const double priorDuty   = ta->getPriorDuty();       // 'NaN' on step 0

// reporting
std::ostringstream oss1;
oss1 << ceilingDuty << " " << priorDuty << "\n";
s_logger->repx(logga::dbug, "current ceiling duty, prior duty", oss1.str());

// undertake adaptation: first skip step 0, then screen on
// hysteresis band (if any), then adjust the bid prices by
// first multiplying and then adding or subtracting

std::string process = "(coding error)";
if ( d_step == 0 )
  {
    // no adaptation necessary or possible
    process = "first pass with initial bidset";
    s_logger->repx(logga::adhc, process, "");
  }
else if ( priorDuty < d_targetCommitment * (1.0 - d_relativeHysteresis) )
  {
    // CAUTION: dereferencing of 'd_bidset' is necessary
    *d_bidset *= (1.0 - d_priceFactor);      // decrease prices
    *d_bidset -= d_priceDelta;               // decrease prices
    process = "bidset price decreased adaptively";
    s_logger->repx(logga::adhc, process, "");
  }
else if ( priorDuty > d_targetCommitment * (1.0 + d_relativeHysteresis) )
  {
    *d_bidset *= (1.0 + d_priceFactor);      // increase prices
    *d_bidset += d_priceDelta;               // increase prices
    process = "bidset price increased adaptively";
    s_logger->repx(logga::adhc, process, "");
 }
else
  {
    process = "bidset remains unchanged";
    s_logger->repx(logga::adhc, process, "");
    // nothing to do
  }

// check for asset capacitation through insufficient bidding
const double hiQuantity = d_bidset->getHiQuantity();
if ( ceilingDuty > hiQuantity )
  {
    std::ostringstream oss2;
    oss2 << ceilingDuty << " : " << hiQuantity;
    s_logger->repx(logga::rankJumpy, "asset capacitation, asset : bid", oss2.str());
  }

// OSP upload bidset call
int ctlDutyGol;
boost::tie(ctlDutyGol) = d_ctl->uploadBidSet(d_bidset);

// OSP couple call (from unit 'b/optprob')
xeona::couple(d_solver,
              opsDutyGol,
              ctlDutyGol,
              lab.str("couple"));

// store bids
d_bidsets.push_back(d_bidset);

// store some information
d_ctls.push_back(d_ctl);

// store bidset in string form
const std::string bidsetstr    = d_bidset->stringify();
d_submittedBidsets->at(d_step) = bidsetstr;
s_logger->repx(logga::adhc, "bidset as string", bidsetstr);

// additional reporting as appropriate
// YEEK 17 CODE (set by '--yeek')
if ( xeona::yeek == 17 || xeona::yeek == 1 )
  {
    std::string priorstr = "(no prior bidset)";
    if ( d_step > 0 ) priorstr = d_submittedBidsets->at(d_step - 1);
    const double weightedPrice = d_bidset->getWeightedPrice();
    std::ostringstream put;
```

```
          put << "  yeek 17 reporting"                                    << "\n"
              << "    function               : AsopLmpBidAdaptive1::" << __func__ << "\n"
              << "    step                   : " << d_step                << "\n"
              << "    bidset pointer         : " << d_bidset              << "\n"
              << "    OSP pointer            : " << d_ctl                 << "\n"
              << "    asset operator  id     : " << aoid                  << "\n"
              << "    technical asset id     : " << taid                  << "\n"
              << "    ops OSP duty gol       : " << opsDutyGol            << "\n"
              << "    ctl OPS duty gol       : " << ctlDutyGol            << "\n"
              << "    current ceiling duty   : " << ceilingDuty           << "\n"
              << "    prior actual duty      : " << priorDuty             << "\n"
              << "    hi bound               : " << hiQuantity            << "\n"
              << "    prior bidset string    : " << priorstr              << "\n"
              << "    current bidset string  : " << bidsetstr             << "\n"
              << "    weighted average price : " << weightedPrice         << "\n"
              << "    process                : " << process               << "\n";
          s_logger->repx(logga::yeek, "additional reporting follows, yeek", xeona::yeek);
          s_logger->putx(logga::yeek, put);
          s_logger->addSmartBlank(logga::yeek);
        }

    } // technical assets loop

  s_logger->repx(logga::dbug, "leaving member function, teas loops", teasLoops);
  // return
  return teasLoops;

} // function 'AsopLmpBidAdaptive1::constrain'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asop03.cc
//  file-create-date : Thu 26-Nov-2009 13:22 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete asset operators 3 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/asop03.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "asop03.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../b/teas.h"        // technical asset entity
#include "../b/optctl.h"      // control optimization sub-problems for asset operators
#include "../b/node.h"        // LMP node entity
#include "../b/lmpbid.h"      // LMP auction bidset
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/bandtaf.h"     // banded tariff set and support
#include "../a/exent.h"       // entity exception classes

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ----------------------------------------------------------
//  CLASS           : AsopInelasticTs
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopInelasticTs
// ----------------------------------------------------------

AsopInelasticTs::AsopInelasticTs
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId, record, xeona::e_commitmentModes),
  d_demands(record.tieTimeseries<double>("demands")),
  d_ctl(),                                    // empty shared pointer
  d_ctls()                                    // empty vector
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
```

```
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopInelasticTs
// ----------------------------------------------------------

AsopInelasticTs::~AsopInelasticTs()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// DOMAIN CONTROLLER POINTS OF ENTRY

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling
//  Status       : complete but not tested
// ----------------------------------------------------------

const int                          // number of technical assets processed
AsopInelasticTs::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invocation produces a new set of OSPs
  d_ctls.clear();                          // strictly necessary

  // demand value
  const double demand = d_demands->at(d_step);

  // declare some administration counters
  int teasLoops = 0;                       // number of technical assets processed

  // step thru technical assets in no required order
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoops;
      xeona::putxId(ta, "AsopInelasticTs::constrain technical asset loop");

      // create and fill a label object
      Label lab(asopId);

      // constrain the associated technical asset
      const int opsDutyGol = ta->constrain(capacityMode);

      // bid coupling is required
      if ( opsDutyGol == 0 )
        {
          const std::string teasId = ta->getIdentifier();
          s_logger->repx(logga::warn, "hollow coupling encountered, gol", opsDutyGol);
          s_logger->repx(logga::xtra, "abandoning current asset processing", teasId);
          continue;
        }

      // recreate and label new control OSP of the required type
      d_ctl.reset(new CtlQuan(d_solver, d_commitmentMode));
      d_ctl->loadOspLabel(lab.str("inelastic-ts"));

      // upload demand quantity
      int ctlDutyGol = -1;                 // nonsensical value
      boost::tie(ctlDutyGol) = d_ctl->uploadControl(demand);

      // OSP couple call (from unit 'b/optprob')
      xeona::couple(d_solver,
                    opsDutyGol,
                    ctlDutyGol,
                    lab.str("couple"));

      // store some information
      d_ctls.push_back(d_ctl);
```

```
    } // technical assets loop

  // return combined count
  return teasLoops;

} // function 'constrain'

// ----------------------------------------------------------
//  CLASS           : AsopAdaptiveTs
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : AsopAdaptiveTs
// ----------------------------------------------------------

AsopAdaptiveTs::AsopAdaptiveTs
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  AssetOperator(entityId, record, xeona::e_commitmentModes),
  d_demands(record.tieTimeseries<double>("demands")),
  d_unitPriceThreshold(record.tieSingle<double>("unit-price-threshold")),
  d_adaptFactor(record.tieSingle<double>("adapt-factor")),
  d_curtailments(record.tieTimeseries<bool>("curtailments")),
  d_ctl(),                                  // empty shared pointer
  d_ctls()                                  // empty vector
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";

  // integrity checks
  if ( d_adaptFactor < 0.0 )
    {
      s_logger->repx(logga::warn, "negative adapt factor", d_adaptFactor);
    }
  else if ( d_adaptFactor > 1.0 )
    {
      s_logger->repx(logga::info, "above unity adapt factor", d_adaptFactor);
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~AsopAdaptiveTs
// ----------------------------------------------------------

AsopAdaptiveTs::~AsopAdaptiveTs()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// DOMAIN CONTROLLER POINTS OF ENTRY

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : constrain call
//  Role         : standard call
//  Techniques   : duty coupling
//  Status       : complete but not tested
// ----------------------------------------------------------

const int                          // number of technical assets processed
AsopAdaptiveTs::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // preamble
  const std::string asopId = getIdentifier();

  // clear previous OSPs because each invocation produces a new set of OSPs
  d_ctls.clear();                           // strictly necessary

  // demand value
  double demand = d_demands->at(d_step);

  // declare some administration counters
  int teasLoops = 0;                        // number of technical assets processed

  // step thru technical assets in no required order
```

```
  BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, d_technicalAssets )
    {
      // keep count
      ++teasLoops;
      xeona::putxId(ta, "AsopAdaptiveTs::constrain technical asset loop");

      // adaptive behavior
      adapt1(ta, demand);

      // create and fill a label object
      Label lab(asopId);

      // constrain the associated technical asset
      const int opsDutyGol = ta->constrain(capacityMode);

      // bid coupling is required
      if ( opsDutyGol == 0 )
        {
          const std::string teasId = ta->getIdentifier();
          s_logger->repx(logga::warn, "hollow coupling encountered, gol", opsDutyGol);
          s_logger->repx(logga::xtra, "abandoning current asset processing", teasId);
          continue;
        }

      // recreate and label new control OSP of the required type
      d_ctl.reset(new CtlQuan(d_solver, d_commitmentMode));
      d_ctl->loadOspLabel(lab.str("inelastic-ts"));

      // upload demand quantity
      int ctlDutyGol = -1;                    // nonsensical value
      boost::tie(ctlDutyGol) = d_ctl->uploadControl(demand);

      // OSP couple call (from unit 'b/optprob')
      xeona::couple(d_solver,
                    opsDutyGol,
                    ctlDutyGol,
                    lab.str("couple"));

      // store some information
      d_ctls.push_back(d_ctl);

    } // technical assets loop

  // return combined count
  return teasLoops;

} // function 'constrain'

// ------------------------------------------------------------
//  MEMBER FUNCTION : adapt1
// ------------------------------------------------------------

bool
AsopAdaptiveTs::adapt1
(shared_ptr<TechnicalAsset> ta,
 double&                    demand)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // preamble
  const double old = demand;
  bool curtailment = false;

  // adaptive behavior
  shared_ptr<BandedTariffSet> tariffset = ta->obtainTariffSet();
  const double marginalPrice            = tariffset->getMarginalPrice(demand);
  if ( marginalPrice > d_unitPriceThreshold )
    {
      // modify current demand
      curtailment = true;
      demand *= d_adaptFactor;
    }
  else
    {
      curtailment = false;
    }

  // update curtailments data
  d_curtailments->at(d_step) = curtailment;
```

```
  // additional reporting as appropriate
  // YEEK 35 CODE (set by '--yeek')
  if ( xeona::yeek == 35 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  curtailment : " << curtailment << "\n"
          << "  demand old  : " << old         << "\n"
          << "  demand new  : " << demand      << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // return
  return curtailment;

} // function 'adapt1'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : auxs.cc
//  file-create-date : Thu 16-Jul-2009 21:49 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : classes for auxiliary model data / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/auxs01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "auxs01.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../b/lmpbid.h"      // LMP auction bidset

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : AuxBidSets
// ----------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger AuxBidSets::s_logger = logga::ptrLogStream();

// CREATORS

AuxBidSets::AuxBidSets
(Record& record):
  d_lmpBidsets(record.tieTimeseries<std::string>("lmp-bidsets"))
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

AuxBidSets::~AuxBidSets()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ACCESSORS (INTERPRETED RETURN)

const shared_ptr<LmpBidSet>
AuxBidSets::getBidSet
```

```
(const int step) const
{
  // if need be, could place the 'at' call in a try block and
  // then return an empty pointer on failure

  const std::string sBidset = d_lmpBidsets->at(step);
  shared_ptr<LmpBidSet> bidset(new LmpBidSet("auxbidset"));
  if ( bidset->pushString(sBidset) != 1 )
    {
      s_logger->repx(logga::warn, "bidset string parse issue", sBidset);
    }
  return bidset;
}

// ----------------------------------------------------------
//  CLASS           : AuxHeatLead
// ----------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger AuxHeatLead::s_hl_logger = logga::ptrLogStream();

// CREATORS

AuxHeatLead::AuxHeatLead
(Record& record):
  d_cogenHeatLeadWeight(record.tieSingle<double>("cogen-heat-lead-weighting"))
{
  s_hl_logger->repx(logga::xtra, "constructor call", "");

  // range checking
  if ( d_cogenHeatLeadWeight < 0.0 || d_cogenHeatLeadWeight > 1.0 )
    {
      const double dcog = d_cogenHeatLeadWeight;
      s_hl_logger->repx(logga::warn, "cogen-heat-lead-weighting not [0,1]", dcog);
    }
}

AuxHeatLead::~AuxHeatLead()
{
  s_hl_logger->repx(logga::adhc, "destructor call", "");
}

// ACCESSORS

const double
AuxHeatLead::getCogenHeatLeadWeight() const
{
  return d_cogenHeatLeadWeight;
}

// MANIPULATORS

void
AuxHeatLead::setCogenHeatLeadWeight
(const double cogenHeatLeadWeight)
{
  s_hl_logger->repx(logga::adhc, "entering member function, weight", cogenHeatLeadWeight);

  // integrity checks
  if ( cogenHeatLeadWeight < 0.0 || cogenHeatLeadWeight > 1.0 )
    {
      s_hl_logger->repx(logga::warn, "weighting out of range [0,1]", cogenHeatLeadWeight);
      s_hl_logger->repx(logga::dbug, "retaining prior value", d_cogenHeatLeadWeight);
      return;
    }

  // update
  d_cogenHeatLeadWeight = cogenHeatLeadWeight;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : bandtaf.cc
//  file-create-date : Tue 18-Nov-2008 09:37 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : banded tariff set and support / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/bandtaf.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "bandtaf.h"          // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <numeric>            // STL numerical algorithms
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <utility>            // STL pair, make_pair()
#include <vector>             // STL sequence container

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

//  CODE

// ----------------------------------------------------------
//  FREE FUNCTION   : std::pair<double, double> ::operator+
// ----------------------------------------------------------
//  Description  : 'std::pair<double, double>' addition
//  Role         : required by 'std::accumulate'
//  Techniques   : free function, operator overloading
//  Status       : complete
//
//  CAUTION: file scope
//
//      This operator is currently set in file scope, but that
//      could be relaxed if deemed useful.  The given semantics
//      should meet most expectations.
//
// ----------------------------------------------------------

namespace
{
  const std::pair<double, double>
  operator+
  (const std::pair<double, double>& lhs,
```

```
   const std::pair<double, double>& rhs)
  {
    std::pair<double, double> tmp(lhs.first  + rhs.first,
                                  lhs.second + rhs.second);
    return tmp;
  }
}

// ----------------------------------------------------------
//  FUNCTION        : ::tariff_less
// ----------------------------------------------------------
//  Description  : custom STL comparison predicate
//  Role         : for 'std::max_element' and similar
//  Techniques   : STL algorithms, unnamed namespace (file local scope)
//  Status       : working
// ----------------------------------------------------------

namespace
{
  bool
  tariff_less
  (std::pair<double, double> one,
   std::pair<double, double> two)
  {
    return one.second < two.second;          // compares second element
  }
} // unnamed namespace

// ----------------------------------------------------------
//  CLASS           : BandedTariffSet
// ----------------------------------------------------------
//  Description  : contract-to-supply tariff set abstraction
//  Role         : used to prepare, hold, and unpack a variety of banded tariff sets
//  Techniques   : stand-alone class, 'std::pair', 'std::stable_sort'
//  Status       : complete
//
//  Design notes
//
//      Overview
//
//          A banded tariff is an UNSORTED collection of (band,
//          price) 'std::pair's held in a 'std::vector.'  Unlike
//          an LMP bidset, the ORDER OF INSERTION is significant.
//
//          In this file, the term 'price' more particularly
//          means 'unit price' in [CUR/J].  A band is specified
//          in [W].
//
//          A 'specific fixed charge' in [CUR/Ws] may also be
//          specified, otherwise the hardcoded default of zero
//          will be used (note that "Ws" is Watt x second).
//
//          A given 'unit price' and 'specific fixed charge' may
//          be strictly positive, zero, or strictly negative.  A
//          'band' must be strictly positive and any
//          non-complying 'band' will be skipped during parsing.
//
//      Input format
//
//          In terms of input, the 'xeona' model more generally
//          expects:
//
//            * a commodity band in [W]
//            * a unit price in [CUR/J]
//
//          In addition, a fixed charge can be given.  If not,
//          the hardcoded default of zero will be used:
//
//            * a specific fixed charge in [CUR/Ws]
//              (or equivalently [CUR/s/W])
//
//          Moreover, the fixed charge can only be given once.
//          Any additional specification is skipped and a warning
//          is logged.
//
//          The 'load' call expects a single string with the
//          following formatting rules:
//
//            * tariff pair as given above: (band, price)
//            * special case single field: fixed charge
//            * tariff separator as defined by 'xeona::modelBidDelim'
```

```
//            * band/price separator is space character
//
//          The individual tariffs are inserted in the order given.
//
//          String insertion is supported via 'pushString' in the
//          prescribed format, for instance:
//
//              "8.00e-6 * 40.00e+06 28.00e-09 * 20.00e+06 40.00e-09"
//
//      Interpretation
//
//          The following interpretations are supported under
//          'summarizeAll':
//
//              "null data or not loaded"
//              "free supply"
//              "simple"
//              "take-or-pay"
//              "standard"
//
// ----------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger BandedTariffSet::s_logger = logga::ptrLogStream();

// CREATORS

BandedTariffSet::BandedTariffSet
(std::string label) :
  d_label(label),
  d_fixed(0.0),
  d_tariffset_pop(),
  d_tariffset_fix(),
  d_capacity(std::numeric_limits<double>::infinity())
{
  s_logger->repx(logga::xtra, "constructor call, label", d_label);
}

BandedTariffSet::~BandedTariffSet()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// MANIPULATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : clear
// ----------------------------------------------------------

void
BandedTariffSet::clear()                    // remove tariffs and zero the fixed charge
{
  s_logger->repx(logga::adhc, "entering member function", "");
  d_tariffset_pop.clear();
  d_tariffset_fix.clear();
  d_fixed = 0.0;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : setLabel
// ----------------------------------------------------------

std::string                                 // return old label
BandedTariffSet::setLabel                   // set new label
(const std::string& label)
{
  std::string temp = d_label;
  d_label = label;
  return temp;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : pushString
// ----------------------------------------------------------
//  Description  : tariffset parsing code for say "50 * 40.00e+06 28.00e-09"
//  Role         : loading a 'BandedTariffSet' instance using model input data
//  Techniques   : Boost.String_algo, calls 'pushTariff' for insertion in 'd_tariffset_*'
//  Status       : complete
//
//  Design notes
```

```
//
//      As currently coded, this member function can be called
//      multiple times.
//
//      This code could have used the Boost.Tokenizer library,
//      but 'boost::algorithm::split' from the Boost.String_algo
//      library is employed instead.
//
//      The "sName" prefix is used to indicate a string variable.
//
//      Note that 'continue' and 'break' are supported by the
//      'Boost.Foreach' library.
//
// ------------------------------------------------------

int                                         // number of tariffs loaded this time
BandedTariffSet::pushString                 // parses input, uses 'pushTariff' to load
(const std::string sTariffset)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, strlen", sTariffset.length());
  int loopCount  = 0;                       // also the return value
  int fixedCount = 0;

  // defensive programming
  if ( sTariffset.empty() )
    {
      s_logger->repx(logga::warn, "empty tariff set string", "");
      return loopCount;
    }

  // note: insert 'clear' call here to reset each time

  // split input string into tariffs
  std::vector<std::string> sTariffs;              // split on tariff separator
  const std::string bs = xeona::modelBidDelim;    // bid separator set in 'common.cc'
  boost::algorithm::split(sTariffs,
                          sTariffset,
                          boost::algorithm::is_any_of(bs),  // tariff separator
                          boost::algorithm::token_compress_off);

  // process individual tariffs
  std::vector<std::string> fields;          // split on price-quantity separator
  double fixed;                             // double for assignment
  tariff_type tariff;                       // tariff for insertion
  BOOST_FOREACH( std::string sTariff, sTariffs )
    {
      // split individual tariffs into fields
      fields.clear();                       // remove all elements
      boost::algorithm::trim(sTariff);      // CAUTION: 'token_compress_on' insufficient
      boost::algorithm::split(fields,
                              sTariff,
                              boost::algorithm::is_any_of(" "),  // field separator
                              boost::algorithm::token_compress_on);
      // switch on structure
      const int elementCount = fields.size();
      switch ( elementCount )
        {
        case 1:
          // attempt a string to double cast
          try
            {
              fixed = boost::lexical_cast<double>(fields.at(0));
            }
          catch( const boost::bad_lexical_cast& eblc)
            {
              // what "bad lexical cast: source type value could
              // not be interpreted as target"
              std::ostringstream put;
              put << "  " << "tariff set string to double cast failure" << "\n"
                  << "  " << "split string: " << fields.at(0) << "\n"
                  << "  " << "what: " << eblc.what() << "\n";
              s_logger->putx(logga::warn, put);
              continue;
            }
          // load data
          ++fixedCount;
          if ( fixedCount == 1 )            // load data if first time
            {
              setSpecificFixedCharge(fixed); // member function call
            }
```

```
          else                                  // more than one fixed charges present
            {
              std::ostringstream oss;
              oss << fixedCount << " " << fixed;
              s_logger->repx(logga::warn, "additional fixed charge ignored", oss.str());
            }
          break;

        case 2:
          // attempt a string to double cast
          try
            {
              tariff.first  = boost::lexical_cast<double>(fields.at(0));
              tariff.second = boost::lexical_cast<double>(fields.at(1));
            }
          catch( const boost::bad_lexical_cast& eblc)
            {
              // what "bad lexical cast: source type value could
              // not be interpreted as target"
              std::ostringstream put;
              put << "  " << "tariff set string to double cast failure" << "\n"
                  << "  " << "split strings: " << fields.at(0)
                  << " " << fields.at(1) << "\n"
                  << "  " << "what: " << eblc.what() << "\n";
              s_logger->putx(logga::warn, put);
              continue;
            }
          // load data
          if ( pushTariff(tariff) > 0 )      // member function for sorted insertion
            {
              ++loopCount;                   // zero indicates failure to insert
            }
          break;

        default:
          s_logger->repx(logga::warn, "illegal tariff field size", elementCount);
          break;
        }
    }
  return loopCount;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : pushTariff
// ------------------------------------------------------------
//  Description  : addes a new tariff
//  Role         : direct use or via 'pushString'
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

int                                         // current number of tariffs or zero on fail
BandedTariffSet::pushTariff
(const tariff_type tariff)
{
  // initial reporting
  std::ostringstream oss;
  oss << tariff.first << " " << tariff.second;
  s_logger->repx(logga::adhc, "entering member function, tariff", oss.str());

  // integrity checks
  if ( tariff.first < 0.0 )
    {
      s_logger->repx(logga::warn, "negative tariff band unacceptable", tariff.first);
      return 0;
    }
  else if ( tariff.first == 0.0 )
    {
      s_logger->repx(logga::dbug, "zero tariff band skipped", tariff.first);
      return 0;
    }

  // load the tariff
  d_tariffset_pop.push_back(tariff);         // unsorted insertion
  d_tariffset_fix.push_back(tariff);         // unsorted insertion
  return d_tariffset_fix.size();             // excludes any popping
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : setSpecificFixedCharge
// ------------------------------------------------------------
```

```
void
BandedTariffSet::setSpecificFixedCharge
(const double specFixedCharge)
{
  d_fixed = specFixedCharge;                // overwrite action
}

// ------------------------------------------------------
//  MEMBER FUNCTION : popLast
// ------------------------------------------------------

BandedTariffSet::tariff_type
BandedTariffSet::popLast()                  // pop newest tariff
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( d_tariffset_pop.empty() )
    {
      s_logger->repx(logga::warn,
                     "attempt to pop an empty tariff set",
                     d_tariffset_pop.size());
      return tariff_type(0.0, 0.0);
    }

  // active code
  tariff_type temp;
  temp = d_tariffset_pop.back();            // CAUTION: 'back' requires element to exist
  d_tariffset_pop.pop_back();               // CAUTION: 'pop_back' doesn't return element
  return temp;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : popFirst
// ------------------------------------------------------

BandedTariffSet::tariff_type
BandedTariffSet::popFirst()                 // pop oldest tariff
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( d_tariffset_pop.empty() )
    {
      s_logger->repx(logga::warn,
                     "attempt to pop an empty tariff set",
                     d_tariffset_pop.size());
      return tariff_type(0.0, 0.0);
    }

  tariff_type temp;
  temp = d_tariffset_pop.front();                 // CAUTION: 'front' element must exist
  d_tariffset_pop.erase(d_tariffset_pop.begin()); // CAUTION: 'erase' element must exist
  return temp;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : truncate
// ------------------------------------------------------

bool                                        // 'true' indicates the 'capacity' binds
BandedTariffSet::truncate
(const double capacity)
{
  // initial reporting
  s_logger->repx(logga::adhc, "leaving member function, capacity", capacity);

  // defensive programming
  if ( capacity < 0.0 )
    {
      s_logger->repx(logga::warn, "negative capacity rejected", capacity);
      return false;
    }

  // active code
  if ( capacity > getBandSum() )
    {
      s_logger->repx(logga::dbug, "capacity is non-binding", capacity);
```

```
      d_capacity = capacity;
      return false;
    }
  else
    {
      s_logger->repx(logga::dbug, "capacity is non-binding", capacity);
      d_capacity = capacity;
      return true;
    }
}

// ACCESSORS

// -------------------------------------------------------
//  MEMBER FUNCTION : getLabel
// -------------------------------------------------------

std::string
BandedTariffSet::getLabel() const
{
  return d_label;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getSpecificFixedCharge
// -------------------------------------------------------

double
BandedTariffSet::getSpecificFixedCharge() const
{
  return d_fixed;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getFirstOnTariff
// -------------------------------------------------------

BandedTariffSet::tariff_type
BandedTariffSet::getFirstOnTariff() const
{
  if ( d_tariffset_fix.size() == 0 )
    {
      return tariff_type(0.0, 0.0);
    }
  else
    {
      return d_tariffset_fix.front();
    }
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getLastOnTariff
// -------------------------------------------------------

BandedTariffSet::tariff_type
BandedTariffSet::getLastOnTariff() const
{
  if ( d_tariffset_fix.size() == 0 )
    {
      return tariff_type(0.0, 0.0);
    }
  else
    {
      return d_tariffset_fix.back();
    }
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getHighestTariff
// -------------------------------------------------------

BandedTariffSet::tariff_type
BandedTariffSet::getHighestTariff() const
{
  if ( d_tariffset_fix.size() == 0 )
    {
      return tariff_type(0.0, 0.0);
    }
  else
    {
      return *std::max_element(d_tariffset_fix.begin(),     // refer <algorithm>
```

```
                                d_tariffset_fix.end(),
                                &::tariff_less);             // refer this file
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLowestTariff
// ----------------------------------------------------------

BandedTariffSet::tariff_type
BandedTariffSet::getLowestTariff() const
{
  if ( d_tariffset_fix.size() == 0 )
    {
      return tariff_type(0.0, 0.0);
    }
  else
    {
      return *std::min_element(d_tariffset_fix.begin(),     // refer <algorithm>
                                d_tariffset_fix.end(),
                                &::tariff_less);             // refer this file
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getBandSum
// ----------------------------------------------------------

double
BandedTariffSet::getBandSum() const
{
  tariff_type initialValue(0.0, 0.0);
  tariff_type sum = std::accumulate(d_tariffset_fix.begin(),
                                    d_tariffset_fix.end(),
                                    initialValue,
                                    &::operator+);     // CAUTION: arg must be explicit
  return sum.first;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getCapacity
// ----------------------------------------------------------

double                                          // could be infinity
BandedTariffSet::getCapacity() const
{
  return d_capacity;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : isNonConvex
// ----------------------------------------------------------

// design allows for strictly negative prices

bool
BandedTariffSet::isNonConvex() const
{
  // shuffle thru and check for a following lower unit price

  const double bigneg = -std::numeric_limits<double>::max();
  tariff_type comp(0.0, bigneg);             // only second element is used
  BOOST_FOREACH( tariff_type t, d_tariffset_fix )
    {
      if ( t.second < comp.second )          // strictly less
        {
          return true;
        }
      comp = t;
    }
  return false;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : isConvex
// ----------------------------------------------------------

bool
BandedTariffSet::isConvex() const
{
  return ( isNonConvex() == false );
```

```
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : empty
// ----------------------------------------------------------

bool
BandedTariffSet::empty() const
{
  return ( d_tariffset_pop.size() == 0 );
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : unfilled
// ----------------------------------------------------------

bool
BandedTariffSet::unfilled() const            // default fixed charge and no tariff data
{
  return ( d_fixed == 0.0                    // default value
           &&
           d_tariffset_fix.size() == 0 );    // no tariff data
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : size
// ----------------------------------------------------------

int                                          // current number of tariffs
BandedTariffSet::size() const
{
  return  d_tariffset_pop.size();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : bands
// ----------------------------------------------------------

int                                          // current number of tariffs
BandedTariffSet::bands() const
{
  return  d_tariffset_fix.size();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : summarizeAll
// ----------------------------------------------------------
//  Description  : output 'd_tariffset_fix' in human readable form
//  Role         : development purposes
//  Techniques   : 'Boost.Format'
//  Status       : complete
//
//  Design notes
//
//      Typical output
//
//          The following is typical.  The first part has
//          suppressed precision, whereas the the part in
//          brackets does not.  The currency string is now "$".
//
//    tariff set label: for-unit-testing
//       4.40e-00 $/s  (4.4)
//       40.00e+06 W  28.00e-09 $/J  (4e+07, 2.8e-08)
//       30.00e+06 W  10.00e-09 $/J  (3e+07, 1e-08)
//       20.00e+06 W  80.00e-09 $/J  (2e+07, 8e-08)
//       3 bands totaling: 90.00e+06 W
//       price range: 10.00e-09 - 80.00e-09 $/J  first/last: 28.00e-09, 80.00e-09 $/J
//       summary: non-convex (at least once) with nonzero fixed charge
//
// ----------------------------------------------------------

std::string
BandedTariffSet::summarizeAll() const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  std::ostringstream oss;                    // for return as string

  // for convenience
  const int bandz = bands();
```

```
// summarize label
std::string tag = getLabel();
if ( tag.empty() ) tag = "(not set)";
oss << "  " << "tariff set label: " << tag << "\n"
    << "  " << "  loaded data"             << "\n";

// summarized fixed
oss << "  " << "   " << boost::format("% .2fe-00 $/s") % d_fixed
    << "  " << "(" << d_fixed << ")" << "\n";

// loop and summarize tariffs
// CAUTION: BOOST_FOREACH requires typedef
BOOST_FOREACH( tariff_type tariff, d_tariffset_fix )
  {
    oss << "  "
        << "   "
        << boost::format("% 6.2fe+06 W")   % (tariff.first  / 1e6)
        << " "
        << boost::format("% 6.2fe-09 $/J") % (tariff.second * 1e9)
        << "  " << "(" << tariff.first << ", " << tariff.second << ")"
        << "\n";
  }

// add subheading
oss << "    " << "summary" << "\n";

// summarize band sum
if ( bandz == 1 ) oss << "      " << "1 band";
else              oss << "      " << bandz << " bands";
oss << " totaling:" << boost::format("% .2fe+06 W") % (getBandSum() / 1e6)
    << "\n";

// summarize prices
oss << "      " << "price range:"
    << boost::format("% .2fe-09")     % (getLowestTariff().second  * 1e9)
    << " to"
    << boost::format("% .2fe-09 $/J") % (getHighestTariff().second * 1e9)
    << "  " << "first/last:"
    << boost::format("% .2fe-09")     % (getFirstOnTariff().second * 1e9)
    << " and"
    << boost::format("% .2fe-09 $/J") % (getLastOnTariff().second  * 1e9)
    << "\n";

// interpret the tariff set
std::string sy;

if ( getSpecificFixedCharge() == 0.0 )     // with zero fixed charge
  if      ( bandz == 0 )  sy = "empty with zero fixed charge (null data or not loaded)";
  else if ( bandz == 1 )
    if ( getHighestTariff().second == 0.0 )
                          sy = "zero price with zero fixed charge "
                            "(\"free supply\")";
    else                  sy = "single (non-banded) with zero fixed charge "
                            "(a \"simple\" contract)";
  else
    if ( isNonConvex() )  sy = "non-convex (at least once) with zero fixed charge";
    else                  sy = "convex (never decreasing) with zero fixed charge";
else if ( getSpecificFixedCharge() > 0.0 ) // with positive fixed charge
  if      ( bandz == 0 )  sy = "empty with nonzero fixed charge";
  else if ( bandz == 1 )
    if ( getHighestTariff().second == 0.0 )
                          sy = "zero price with nonzero fixed charge "
                            "(a \"take-or-pay\" contract)";
    else                  sy = "single (non-banded) with nonzero fixed charge"
                            " (a \"standard\" contract)";
  else
    if ( isNonConvex() )  sy = "non-convex (at least once) with nonzero fixed charge";
    else                  sy = "convex (never decreasing) with nonzero fixed charge";
else if ( getSpecificFixedCharge() < 0.0 ) // with negative fixed charge
  if      ( bandz == 0 )  sy = "empty with negative fixed charge";
  else if ( bandz == 1 )
    if ( getHighestTariff().second == 0.0 )
                          sy = "zero price with negative fixed charge";
    else                  sy = "single (non-banded) with negative fixed charge";
  else
    if ( isNonConvex() )  sy = "non-convex (at least once) with negative fixed charge";
    else                  sy = "convex (never decreasing) with negative fixed charge";
else                                       // should not be here
  std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
```

```
  oss << "      " << "contract type: "<< sy << "\n";

  return oss.str();

} // member function 'BandedTariffSet::summarizeAll'

// -----------------------------------------------------------
//  MEMBER FUNCTION : getMarginalPrice
// -----------------------------------------------------------
//  Description  : returns the marginal price for a given quantity
//  Role         : general use
//  Techniques   : simplification of function 'interpretSale'
//  Status       : complete
// -----------------------------------------------------------

double                                      // marginal price ($/quantity)
BandedTariffSet::getMarginalPrice
(const double sale) const                   // transaction (quantity)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // comment on input
  if ( sale == 0)
    {
      s_logger->repx(logga::adhc, "sale is zero", sale);
    }
  else if ( sale < 0 )
    {
      s_logger->repx(logga::warn, "sale is unexpectedly negative", sale);
    }

  // opening state
  double cumulativeBand = 0.0;
  double marginalPrice  = 0.0;
  int    activeTariffs  = 0;

  // loop thru tariffs
  BOOST_FOREACH( tariff_type t, d_tariffset_fix )
    {
      ++activeTariffs;
      double currentBand  = t.first;        // current band
      double currentPrice = t.second;       // current price

      cumulativeBand     += currentBand;    // too high, reduced on loop exit

      if ( sale >= cumulativeBand )         // indicates flooded band
        {
          // loop again
        }
      else                                  // indicates marginal band
        {
          marginalPrice    = currentPrice;  // by definition
          break;                            // bail out
        }
    }

  // return
  return marginalPrice;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : interpretSale
// -----------------------------------------------------------
//  Description  : interprets tariff set in relation to a sale
//  Role         : used by OSP 'OfrTariffSet::downloadSolution'
//  Techniques   : discrete integration
//  Status       : complete
// -----------------------------------------------------------

boost::tuple                                // see 'OfrTariffSet::results_type' typedef
<double,                                    // marginal price ($/quantity)
 double,                                    // fixed component ($)
 double,                                    // total price ($)
 double>                                    // transaction (quantity)
BandedTariffSet::interpretSale
(const double size,
 const double sale,
 const int    interval) const
{
  // initial reporting
```

```
  s_logger->repx(logga::adhc, "entering member function", "");

  // comment on input
  if ( size == 0)
    {
      s_logger->repx(logga::adhc, "size is zero", size);
    }
  else if ( size < 0 )
    {
      s_logger->repx(logga::warn, "size is unexpectedly negative", size);
    }

  // opening state
  double cumulativeBand = 0.0;
  double fixedComponent = 0.0;
  double totalVarCost   = 0.0;               // thru discrete integration
  double marginalPrice  = 0.0;
  double totalCost      = 0.0;
  int    activeTariffs  = 0;

  // fixed component
  fixedComponent = d_fixed * interval;

  // loop thru tariffs
  BOOST_FOREACH( tariff_type t, d_tariffset_fix )
    {
      ++activeTariffs;
      double currentBand  = t.first;         // current band
      double currentPrice = t.second;        // current price

      cumulativeBand     += currentBand;     // too high, reduced on loop exit

      if ( sale >= cumulativeBand )          // indicates flooded band
        {
          totalVarCost    += currentBand * currentPrice;
        }
      else                                   // indicates marginal band
        {
          totalVarCost    += (sale - (cumulativeBand - currentBand)) * currentPrice;
          cumulativeBand   = sale;
          marginalPrice    = currentPrice;   // by definition
          break;                            // bail out
        }
    }

  // total cost
  totalCost = fixedComponent + totalVarCost;

  // completion reporting
  std::ostringstream oss;
  oss << activeTariffs << " of " << d_tariffset_fix.size();
  s_logger->repx(logga::adhc, "complete, active tariffs of all tariffs", oss.str());

  // additional reporting as appropriate
  // YEEK 34 CODE (set by '--yeek')
  if ( xeona::yeek == 34 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << summarizeAll()                  // trailing newline not required
          << ""                                                    << "\n"
          << "  interpret sale"                                    << "\n"
          << "    inputs"                                          << "\n"
          << "      size     : " << size                   << " W"    << "\n"
          << "      sale     : " << sale                   << " J"    << "\n"
          << "      interval : " << interval               << " s"    << "\n"
          << "    return values"                                   << "\n"
          << "      marginal price     : " << marginalPrice  << " $/J"  << "\n"
          << "      fixed component    : " << fixedComponent << " $"    << "\n"
          << "      variable component : " << totalVarCost   << " $"    << "\n"
          << "      total cost         : " << totalCost      << " $"    << "\n"
          << "      transaction        : " << cumulativeBand << " J"    << "\n"
          << ""                                                    << "\n"
          << "  note: this reporting uses J (and W) but should more "
        "generally use . as the generic duty unit and * as the "
        "generic capacity unit"                                    << "\n"
          << "        see 'b/costreg.h' for a discussion on the "
        "treatment and taxonomy of costs"                          << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }
```

```
  // return en-masse
  return boost::make_tuple(marginalPrice,
                           fixedComponent,
                           totalVarCost,
                           cumulativeBand);

} // member function 'BandedTariffSet::interpretSale'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : block.cc
//  file-create-date : Tue 26-Aug-2008 14:21 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : abstract block entity / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/block.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "block.h"            // companion header for this file (place first)

#include "../c/si3units.h"    // support for engineering format and SI prefixes

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting, max()
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting

//  CODE

// CREATORS

// ------------------------------------------------------------
//  MEMBER FUNCTION : Block
// ------------------------------------------------------------

Block::Block
(const std::string entityId,
 Record&           record) :
  FullEntity(entityId, record),
  d_dutyStats(),
  d_sizeStats()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~Block
// ------------------------------------------------------------

Block::~Block()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : getDutyStats
// ----------------------------------------------------------

const Statistics<double>&
Block::getDutyStats() const
{
  return d_dutyStats;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getSizeStats
// ----------------------------------------------------------

const Statistics<double>&
Block::getSizeStats() const
{
  return d_sizeStats;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : formatStats
// ----------------------------------------------------------
//  Description  : line formats current 'duty' and 'size' statistics
//  Role         : end of simulation reporting from 'DomainController::concludeDomain'
//  Techniques   : 'Boost.Format'
//  Status       : complete
//
//  Design notes
//
//      at the time of writing, only called under report level 5
//      or higher
//
//  Input limits (may need to confirm currency)
//
//      identifier : should be 32 chars or less
//      duty/size  : "%10.0f" meaning up from 10GW
//
// ----------------------------------------------------------

const std::string
Block::formatStats
(const int          indent,
 const std::string& trailingComment) const
{
  // process left indent
  const std::string leftIndent(indent, ' ');

  // process trailing comment
  const int noncommentWidth = indent + 118;  // right side of size "count"
  std::string buffer;
  if ( ! trailingComment.empty() )
    {
      const int length = xeona::consoleWidth - noncommentWidth;
      // defensive programming
      if ( length < 0 )
        {
          std::ostringstream oss;
          oss << noncommentWidth << " : " << xeona::consoleWidth;
          s_logger->repx(logga::warn, "console too narrow, width : _XTCOLS", oss.str());
          const std::string msg
            = leftIndent
            + "(console width problem, noncommentWidth : xeona::consoleWidth "
            + oss.str() + ")";
          return msg;
        }
      else                                    // '_XTCOLS' set low
        {
          buffer = "    " + trailingComment;
          buffer = buffer.substr(0, length); // trim
        }
    }

  // integrity checks on sub-block entity programming
  if      ( d_dutyStats.count() == 0 || d_sizeStats.count() == 0 )
    {
      std::ostringstream put;
      put << "  identifier : "            << getIdAndKind()                    << "\n"
          << "  class                : " << getTypeStr()                       << "\n"
          << "  problem              : duty and/or size statistics empty"      << "\n"
          << "  likely cause         : faulty sub-block programming"           << "\n"
```

```
          << "  duty statistics count : " << d_dutyStats.count()               << "\n"
          << "  size statistics count : " << d_sizeStats.count()               << "\n";
      s_logger->repx(logga::rankJumpy,
                     "duty and/or size stats empty",
                     getIdAndKind());
      s_logger->putx(logga::dbug, put);
    }
  else if ( d_dutyStats.count() != d_sizeStats.count() )
    {
      std::ostringstream put;
      put << "  identifier : "           << getIdAndKind()                     << "\n"
          << "  class                : " << getTypeStr()                        << "\n"
          << "  problem              : duty and size statistics counts differ" << "\n"
          << "  likely cause         : faulty sub-block programming"           << "\n"
          << "  duty statistics count : " << d_dutyStats.count()               << "\n"
          << "  size statistics count : " << d_sizeStats.count()               << "\n";
      s_logger->repx(logga::rankJumpy,
                     "duty and size stats counts differ",
                     getIdAndKind());
      s_logger->putx(logga::dbug, put);
    }

  // from commit r4105, the code relies on
  // 'xeona::fmtEngineeringFix' in unit 'c/si3units', this
  // function adds a " ", "k", "M", and so on, while ignoring
  // inf's and nan's

  const double maxDuty = d_dutyStats.max();
  const double maxSize = d_sizeStats.max();

#if 0 // 0 = normal, 1 = more aggressive
  const xeona::SiPrefix prefixDuty = xeona::getEngineeringPrefix(maxDuty);
  const xeona::SiPrefix prefixSize = xeona::getEngineeringPrefix(maxSize);
  const xeona::SiPrefix temp   = std::max(prefixDuty, prefixSize);
  const xeona::SiPrefix prefix = static_cast<xeona::SiPrefix>(temp - 3);
  const unsigned decimals = 2;              // number of decimal places
#else
  const xeona::SiPrefix prefixDuty = xeona::getEngineeringPrefix(maxDuty, 1);
  const xeona::SiPrefix prefixSize = xeona::getEngineeringPrefix(maxSize, 1);
  const xeona::SiPrefix prefix = std::max(prefixDuty, prefixSize);
  const unsigned decimals = 2;              // number of decimal places
#endif // 0

  // note on 'Boost.Format' formatting: '-' is left-align,
  // '10.0f' means ten width, zero decimal places, fixed format
  // (could also try "e") -- obsolete comment now

//const std::string ff    = "   %10.0f";     // CAUTION: any "% " space is significant
  const std::string ff    = " %12s";

  const std::string formstr = "%s%-32s" + ff + ff + ff + ff + ff + ff + "   %5d" + "%s\n";
  return boost::str(boost::format(formstr)
                    % leftIndent
                    % getIdAndKind()
                    % xeona::fmtEngineeringFix(d_dutyStats.min() , prefix, decimals)
                    % xeona::fmtEngineeringFix(d_dutyStats.max() , prefix, decimals)
                    % xeona::fmtEngineeringFix(d_dutyStats.mean(), prefix, decimals)
                    % xeona::fmtEngineeringFix(d_sizeStats.min() , prefix, decimals)
                    % xeona::fmtEngineeringFix(d_sizeStats.max() , prefix, decimals)
                    % xeona::fmtEngineeringFix(d_sizeStats.mean(), prefix, decimals)
                    % d_dutyStats.count()
                    % buffer);
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : formatStatsHeader (static)
// ------------------------------------------------------------

const std::string
Block::formatStatsHeader
(const int indent,
 const int ruleLength)
{
  const std::string leftTab(indent, ' ');
  const std::string rule(ruleLength, '-');
  return leftTab
    + "identifier                      "
    + "      duty lo      duty hi      duty av"
    + "      size lo      size hi      size av"
    + "   count    role"
    + "\n"
```

```
    + leftTab
    + rule
    + "\n";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : formatStatsFooter (static)
// ----------------------------------------------------------

const std::string
Block::formatStatsFooter
(const int indent)
{
  const std::string leftTab(indent, ' ');
  return "\n"
    + leftTab
    + "SI prefixes: "
    + "E exe  P peta  T tera  G giga  M mega  k kilo  --  m milli  u micro  n nano"
    +"\n"
    + leftTab
    + "notes: 'size' is the step-specific UPPER capacity"
    + ", 'inf' indicates no limit"
    + ", '0' means (close-to) zero"
    + ", 'count' is the data count"
    + "\n";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : reportDutyAndSize
// ----------------------------------------------------------

std::string
Block::reportDutyAndSize() const
{
  s_logger->repx(logga::xtra, "entering member function", "");

  std::ostringstream oss;
  oss << "  duty (typically production or sales)"               << "\n"
      << "    current : " << d_dutyStats.last()                 << "\n"
      << "    hi      : " << d_dutyStats.max()                  << "\n"
      << "    lo      : " << d_dutyStats.min()                  << "\n"
      << "  size (usually actual but may be nominal capacity)"  << "\n"
      << "    current : " << d_sizeStats.last()                 << "\n"
      << "    hi      : " << d_sizeStats.max()                  << "\n"
      << "    lo      : " << d_sizeStats.min()                  << "\n";
  return oss.str();

}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : builtins.cc
//  file-create-date : Mon 28-Jan-2008 18:25 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : builtin sub-entities / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/builtins.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "builtins.h"         // companion header for this file (place first)

#include "../c/recset.h"      // record set support
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

//  CODE

// ----------------------------------------------------------
//  CLASS           : TimeHorizon
// ----------------------------------------------------------
//  Description  : time horizon entity
//  Status       : complete
// ----------------------------------------------------------

// CREATORS

TimeHorizon::TimeHorizon
(const std::string entityId,                // enforced unique identifier
 Record&           record) :
  Entity(entityId, record),
  d_builtinRemarkTh(record.tieSingle<std::string>("builtin-remark")),
  d_record(record),                         // for testing purposes
  d_steps(record.tieSingle<int>("steps")),
  d_interval(record.tieSingle<int>("interval")),
  d_startHour(record.tieSingle<int>("start-hour")),
  d_startDay(record.tieSingle<int>("start-day")),
  d_hemisphere(record.tieSingle<std::string>("hemisphere")),
  d_enumHemisphere(xeona::e_notSet)
{
  // initial reporting
  s_logger->repx(logga::dbug, "constructor call", "");

  // range checks
  if ( d_steps < 2 )
    {
      s_logger->repx(logga::warn, "steps not 2 or more", d_steps);
      std::ostringstream put;
      put
        << "  to run just one step, set the 'entity.time-horizon.step' field"    << "\n"
        << "  to 2 or more and then employ the command-line option \"--mode 6\"" << "\n";
```

```
      s_logger->putx(logga::dbug, put);
    }
  if ( d_startHour < 0 || d_startHour >  23 )
    {
      s_logger->repx(logga::warn, "start hour not [0,23]", d_startHour);
    }
  if ( d_startDay  < 1 || d_startDay  > 356 )
    {
      s_logger->repx(logga::warn, "start day not [1,365]", d_startDay);
    }

  // modify
  if      ( d_hemisphere == "N" )
    {
      d_enumHemisphere = xeona::e_north;
    }
  else if ( d_hemisphere == "S" )
    {
      d_enumHemisphere = xeona::e_south;
    }
  else
    {
      s_logger->repx(logga::warn, "hemisphere not {N,S}", d_hemisphere);
    }

  // update the central horizon steps and interval information
  if ( Entity::s_horizonSteps != d_steps )
    {
      std::ostringstream oss;
      oss << Entity::s_horizonSteps << " : " << d_steps;
      s_logger->repx(logga::warn, "steps disagree, existing : mine", oss.str());
    }

  // builtin remark
  d_builtinRemarkTh = "mandatory entity";

  // transfer values
  // CAUTION: these provide copies and not references
  Entity::s_horizonSteps      = d_steps;
  Entity::s_horizonInterval   = d_interval;
  Entity::s_horizonStartHour  = d_startHour;
  Entity::s_horizonStartDay   = d_startDay;
  Entity::s_horizonHemisphere = d_enumHemisphere;
}

TimeHorizon::~TimeHorizon()
{
  s_logger->repx(logga::xtra, "destructor call", "");
}

void
TimeHorizon::factoryInitialize()
{
  s_logger->repx(logga::xtra, "TimeHorizon instance initialization", "");
}

// STREAM INSERTION SUPPORT

std::ostream&
TimeHorizon::streamOut                        // support for overloaded operator<<
(std::ostream& os) const
{
  std::ios_base::fmtflags prior = os.flags();
  os << "    dummy output from a derived entity" << "\n";
  os.flags(prior);
  return os;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : commods.cc
//  file-create-date : Wed 30-Jul-2008 07:37 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : commodities hierarchy / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/commods.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "commods.h"          // companion header for this file (place first)

#include "../c/ghouse.h"      // global warming potential support
#include "../c/extunits.h"    // quantifying extensity enums and interpretation
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : Commodity (abstract)
// ----------------------------------------------------------

Commodity::Commodity
(const std::string          entityId,
 Record&                    record,
 const xeona::ExtensityUnit quantifyingExtensity) :
  FullEntity(entityId, record),
  d_quantifyingExtensity(quantifyingExtensity)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

Commodity::~Commodity()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

int                                     // minus one indicates some problem
Commodity::getInterfacePairs() const
{
  const int interfaces = getUseCount() - 1;  // should be in {0,2,4,..}
  if ( interfaces == -1 )
    {
      s_logger->repx(logga::warn, "commodity not factory constructed", interfaces);
      return -1;
```

```
    }
  // odd/even test
  double check = static_cast<double>(interfaces)/2.0;
  if ( check != std::floor(check) )          // see <cmath>
    {                                        // 'interfaces' was odd
      s_logger->repx(logga::dbug, "interface count not properly paired", interfaces);
      return -1;
    }
  return interfaces / 2;
}

// ----------------------------------------------------------
//  CLASS           : CommodityJoule
// ----------------------------------------------------------

CommodityJoule::CommodityJoule
(const std::string entityId,
 Record&           record) :
  Commodity(entityId, record, xeona::joule)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

CommodityJoule::~CommodityJoule()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

xeona::ExtensityUnit
CommodityJoule::getExtensity() const
{
  return d_quantifyingExtensity;
}

std::string
CommodityJoule::getExtensityStr() const
{
  return xeona::interpretExtensity(d_quantifyingExtensity);
}

// ----------------------------------------------------------
//  CLASS           : CommodityKilogram
// ----------------------------------------------------------

CommodityKilogram::CommodityKilogram
(const std::string entityId,
 Record&           record) :
  Commodity(entityId, record, xeona::kilogram)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

CommodityKilogram::~CommodityKilogram()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

xeona::ExtensityUnit
CommodityKilogram::getExtensity() const
{
  return d_quantifyingExtensity;
}

std::string
CommodityKilogram::getExtensityStr() const
{
  return xeona::interpretExtensity(d_quantifyingExtensity);
}

// ----------------------------------------------------------
//  CLASS           : CommodityUOA
// ----------------------------------------------------------

CommodityUOA::CommodityUOA
(const std::string entityId,
 Record&           record) :
  Commodity(entityId, record, xeona::uoa)
```

```
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

CommodityUOA::~CommodityUOA()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

xeona::ExtensityUnit
CommodityUOA::getExtensity() const
{
  return d_quantifyingExtensity;
}

std::string
CommodityUOA::getExtensityStr() const
{
  return xeona::interpretExtensity(d_quantifyingExtensity);
}

// ----------------------------------------------------------
//  CLASS           : CommodityMetreSq
// ----------------------------------------------------------

CommodityMetreSq::CommodityMetreSq
(const std::string entityId,
 Record&           record) :
  Commodity(entityId, record, xeona::uoa)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

CommodityMetreSq::~CommodityMetreSq()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

xeona::ExtensityUnit
CommodityMetreSq::getExtensity() const
{
  return d_quantifyingExtensity;
}

std::string
CommodityMetreSq::getExtensityStr() const
{
  return xeona::interpretExtensity(d_quantifyingExtensity);
}

// ----------------------------------------------------------
//  CLASS           : CmOxidize (quantified in kg)
// ----------------------------------------------------------

// CREATORS

CmOxidize::CmOxidize
(const std::string entityId,
 Record&           record) :
  CommodityKilogram(entityId, record),
  d_specCombustionEnthalpy(record.tieSingle<double>("spec-combustion-enthalpy")),
  d_specCarbonDioxide(record.tieSingle<double>("spec-carbon-dioxide")),
  d_specCo2equiv(record.tieSingle<double>("spec-co2-equiv"))
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmOxidize::~CmOxidize()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

double
CmOxidize::getSpecCombEnthalpy() const
```

```
{
  return d_specCombustionEnthalpy;
}

double
CmOxidize::getSpecCarbonDioxide() const
{
  return d_specCarbonDioxide;
}

double
CmOxidize::getSpecCo2equiv() const
{
  return d_specCo2equiv;
}

Gwp100Bundle
CmOxidize::getSpecGhgs() const
{
  // initial reporting -- noting that more input data would be
  // needed to implement this function completely
  s_logger->repx(logga::warn, "function not properly implemented", "");

  // active code
  Gwp100Bundle gwps;
  gwps.co2 = d_specCarbonDioxide;
  return gwps;
}

// ----------------------------------------------------------
//  CLASS           : CmCarbonSeq (quantified in kg)
// ----------------------------------------------------------

CmCarbonSeq::CmCarbonSeq
(const std::string entityId,
 Record&           record) :
  CommodityKilogram(entityId, record),
  d_pressures(record.tieTimeseries<double>("pressures"))
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmCarbonSeq::~CmCarbonSeq()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

double
CmCarbonSeq::getPressure
(const int step) const
{
  return d_pressures->at(step);
}

void
CmCarbonSeq::setPressure
(const int    step,
 const double pressure)
{
  d_pressures->at(step) = pressure;
}

// ----------------------------------------------------------
//  CLASS           : CmCarbonCert (quantified in kg)
// ----------------------------------------------------------

CmCarbonCert::CmCarbonCert
(const std::string entityId,
 Record&           record) :
  CommodityKilogram(entityId, record),
  d_unitPrices(record.tieTimeseries<double>("unit-prices"))
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmCarbonCert::~CmCarbonCert()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}
```

```
// -----------------------------------------------------------
//  CLASS           : CmElectricity (quantified in J)
// -----------------------------------------------------------

CmElectricity::CmElectricity
(const std::string entityId,
 Record&           record) :
  CommodityJoule(entityId, record),
  d_voltage(record.tieSingle<double>("voltage"))
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmElectricity::~CmElectricity()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

double
CmElectricity::getVoltage() const
{
  return d_voltage;
}

// -----------------------------------------------------------
//  CLASS           : CmWork (quantified in J)
// -----------------------------------------------------------

CmWork::CmWork
(const std::string entityId,
 Record&           record) :
  CommodityJoule(entityId, record)
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmWork::~CmWork()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// -----------------------------------------------------------
//  CLASS           : CmHeat (quantified in J)
// -----------------------------------------------------------

CmHeat::CmHeat
(const std::string entityId,
 Record&           record) :
  CommodityJoule(entityId, record)
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmHeat::~CmHeat()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// -----------------------------------------------------------
//  CLASS           : CmThermalFluid (quantified in J)
// -----------------------------------------------------------

CmThermalFluid::CmThermalFluid
(const std::string entityId,
 Record&           record) :
  CommodityJoule(entityId, record),
  d_specHeatCapacity(record.tieSingle<double>("spec-heat-capacity")),
  d_floTemps(record.tieTimeseries<double>("flo-temps")),
  d_retTemps(record.tieTimeseries<double>("ret-temps"))
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmThermalFluid::~CmThermalFluid()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
```

```
}

double
CmThermalFluid::getFloTemp
(const int step) const
{
  return d_floTemps->at(step);
}

double
CmThermalFluid::getRetTemp
(const int step) const
{
  return d_retTemps->at(step);
}

void
CmThermalFluid::setFloTemp
(const int    step,
 const double temp)
{
  if ( temp < xeona::absoluteZero )
    {
      s_logger->repx(logga::warn, "flo temp now aphysical", temp);
    }
  d_floTemps->at(step) = temp;
}

void
CmThermalFluid::setRetTemp
(const int    step,
 const double temp)
{
  if ( temp < xeona::absoluteZero )
    {
      s_logger->repx(logga::warn, "ret temp now aphysical", temp);
    }
  d_retTemps->at(step) = temp;
}

double&
CmThermalFluid::floTemp
(const int step)
{
  return d_floTemps->at(step);
}

double&
CmThermalFluid::retTemp
(const int step)
{
  return d_retTemps->at(step);
}

// ------------------------------------------------------------
//  CLASS           : CmFunds (quantified in UOA)
// ------------------------------------------------------------

CmFunds::CmFunds
(const std::string entityId,
 Record&           record) :
  CommodityUOA(entityId, record)
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmFunds::~CmFunds()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  CLASS           : CmFission (quantified in kg)
// ------------------------------------------------------------

CmFission::CmFission
(const std::string entityId,
 Record&           record) :
  CommodityKilogram(entityId, record)
{
```

```
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmFission::~CmFission()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  CLASS           : CmProductiveLand (quantified in m2)
// ----------------------------------------------------------

CmProductiveLand::CmProductiveLand
(const std::string entityId,
 Record&           record) :
  CommodityMetreSq(entityId, record)
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CmProductiveLand::~CmProductiveLand()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : commods01.cc
//  file-create-date : Tue 05-May-2009 18:35 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete commodities 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/commods01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  AD-HOC NOTES
//
//  See r2570 for use of retie code.

//  LOCAL AND SYSTEM INCLUDES

#include "commods01.h"        // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : CmOxidBiocoal
// ----------------------------------------------------------

// CREATORS

CmOxidBiocoal::CmOxidBiocoal
(const std::string entityId,
 Record&           record) :
  CmOxidize(entityId, record)
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());

  if ( d_specCombustionEnthalpy < 10.0e+06 || d_specCombustionEnthalpy > 30.0e+06 )
    {
      s_logger->repx(logga::info,
                     "odd specific combustion enthalpy",
                     d_specCombustionEnthalpy);
    }

  d_builtinRemark = "beta / experimental";
}

// ----------------------------------------------------------
//  CLASS           : ~CmOxidBiocoal
```

```
// ------------------------------------------------------

CmOxidBiocoal::~CmOxidBiocoal()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : costreg.cc
//  file-create-date : Mon 20-Oct-2008 08:57 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : cost registers / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/costreg.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "costreg.h"          // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/fincalc.h"     // support for discounted cash flow analysis
#include "../c/costs.h"       // cost sets and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// STATIC DEFINITIONS

logga::spLogger CostRegister::s_logger_costreg = logga::ptrLogStream();  // bind

// ----------------------------------------------------------
//  CLASS           : CostRegister
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CostRegister
// ----------------------------------------------------------

CostRegister::CostRegister
(Record& record) :
  d_reset(0.0),                             // default set here
  d_interval(Entity::getHorizonInterval()),
  d_horizon(Entity::getHorizonSteps()),
  d_dutySpecCosts(d_reset),
  d_sizeSpecCosts(d_reset),
  d_standingCosts(d_reset),
  d_varFins(new std::vector<double>(d_horizon, d_reset)),
  d_fixFins(new std::vector<double>(d_horizon, d_reset)),
  d_embFins(new std::vector<double>(d_horizon, d_reset)),
  d_varGhgs(new std::vector<double>(d_horizon, d_reset)),
```

```
  d_fixGhgs(new std::vector<double>(d_horizon, d_reset)),
  d_embGhgs(new std::vector<double>(d_horizon, d_reset)),
  d_varNoxs(new std::vector<double>(d_horizon, d_reset)),
  d_fixNoxs(new std::vector<double>(d_horizon, d_reset)),
  d_embNoxs(new std::vector<double>(d_horizon, d_reset)),
  d_varDeps(new std::vector<double>(d_horizon, d_reset)),
  d_fixDeps(new std::vector<double>(d_horizon, d_reset)),
  d_embDeps(new std::vector<double>(d_horizon, d_reset)),
  d_varLucs(new std::vector<double>(d_horizon, d_reset)),
  d_fixLucs(new std::vector<double>(d_horizon, d_reset)),
  d_embLucs(new std::vector<double>(d_horizon, d_reset))

//d_revenues(new std::vector<double>(d_horizon, d_reset))  // intentionally disabled
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call, reset value", d_reset);

  // defensive programming -- worth testing but should never fail
  if ( d_horizon < 2 )
    {
      s_logger_costreg->repx(logga::warn, "horizon problem", d_horizon);
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegister
// ----------------------------------------------------------

CostRegister::~CostRegister()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : updateShortrunCosts (separated form)
// ----------------------------------------------------------

void
CostRegister::updateShortrunCosts
(const int      step,                       // does, in theory, allow for reworking
 const CostSet& var,
 const CostSet& fix)                        // may also contain standing costs
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  d_varFins->at(step) = var.fin * d_interval;
  d_fixFins->at(step) = fix.fin * d_interval;
  d_varGhgs->at(step) = var.ghg * d_interval;
  d_fixGhgs->at(step) = fix.ghg * d_interval;
  d_varNoxs->at(step) = var.nox * d_interval;
  d_fixNoxs->at(step) = fix.nox * d_interval;
  d_varDeps->at(step) = var.dep * d_interval;
  d_fixDeps->at(step) = fix.dep * d_interval;
  d_varLucs->at(step) = var.luc * d_interval;
  d_fixLucs->at(step) = fix.luc * d_interval;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : updateShortrunCosts (tuple wrapper form)
// ----------------------------------------------------------

void
CostRegister::updateShortrunCosts
(const int                              step,
 const boost::tuple<CostSet, CostSet>& shortrun)
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  updateShortrunCosts(step,
                      shortrun.get<0>(),     // CAUTION: zero-based tuple indexing
                      shortrun.get<1>());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : updateEmbeddedCosts (transferred)
// ----------------------------------------------------------
```

```
void
CostRegister::updateEmbeddedCosts
(const int      step,
 const CostSet& emb)
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  d_varFins->at(step) = emb.fin * d_interval;
  d_varGhgs->at(step) = emb.ghg * d_interval;
  d_varNoxs->at(step) = emb.nox * d_interval;
  d_varDeps->at(step) = emb.dep * d_interval;
  d_varLucs->at(step) = emb.luc * d_interval;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : updateEmbeddedCosts (calculated)
// ------------------------------------------------------------

void
CostRegister::updateEmbeddedCosts
(const int step)
{
  // normally overridden so complain if called
  s_logger_costreg->repx(logga::warn, "function is hollow", "");
  std::ostringstream put;
  put << "  function 'CostRegister::updateEmbeddedCosts' (calculated) is hollow"  << "\n"
      << "  try using a more appropriate cost register sub-class for this entity" << "\n";
  s_logger_costreg->putx(logga::adhc, put);
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : resetRegister
// ------------------------------------------------------------
//  Description  : reset all tied standard cost vectors
//  Role         : standard usage
//  Techniques   : 'std::vector' member functions
//  Status       : complete
//
//  Design notes
//
//      The given code tests for tied timeseries and then resets
//      these to 'd_reset' (zero by default).  Note that a prior
//      'clear' call not needed.
//
//      Under g++, the 'std::vector<>::size_type' returned by
//      'std::vector' member function 'size' is 'unsigned'.
//
//      From commit r4913, this call is now considered part of
//      standard usage.  Prior to r4910, it was tagged
//      developmental.
//
//  CAUTION: operates entire horizon
//
//      This function nukes the entire horizon and not just the
//      current 'step'.
//
// ------------------------------------------------------------

void
CostRegister::resetRegister()
{
  s_logger_costreg->repx(logga::dbug, "entering member function, reset to", d_reset);

  // member function 'void std::vector<T>::assign(size_type n,
  // const T& value)' "erases all the elements of the vector,
  // then inserts 'n' copies of 'value'" (Lischner 2003 p724)
  //
  // note also the protection against empty/null shared pointers

  d_varFins->assign(d_varFins->size(), d_reset);
  d_fixFins->assign(d_fixFins->size(), d_reset);
  d_embFins->assign(d_embFins->size(), d_reset);
  d_varGhgs->assign(d_varGhgs->size(), d_reset);
  d_fixGhgs->assign(d_fixGhgs->size(), d_reset);
  d_embGhgs->assign(d_embGhgs->size(), d_reset);
  d_varNoxs->assign(d_varNoxs->size(), d_reset);
  d_fixNoxs->assign(d_fixNoxs->size(), d_reset);
  d_embNoxs->assign(d_embNoxs->size(), d_reset);
  d_varDeps->assign(d_varDeps->size(), d_reset);
```

```
  d_fixDeps->assign(d_fixDeps->size(), d_reset);
  d_embDeps->assign(d_embDeps->size(), d_reset);
  d_varLucs->assign(d_varLucs->size(), d_reset);
  d_fixLucs->assign(d_fixLucs->size(), d_reset);
  d_embLucs->assign(d_embLucs->size(), d_reset);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : importCosts
// ----------------------------------------------------------

void
CostRegister::importCosts
(const int step,
 const boost::tuple
 <CostSet,
  CostSet,
  CostSet>& costs)
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  CostSet var = costs.get<0>();              // CAUTION: zero-based tuple indexing
  CostSet fix = costs.get<1>();
  CostSet emb = costs.get<2>();

  d_varFins->at(step) += var.fin;
  d_fixFins->at(step) += fix.fin;
  d_embFins->at(step) += emb.fin;
  d_varGhgs->at(step) += var.ghg;
  d_fixGhgs->at(step) += fix.ghg;
  d_embGhgs->at(step) += emb.ghg;
  d_varNoxs->at(step) += var.nox;
  d_fixNoxs->at(step) += fix.nox;
  d_embNoxs->at(step) += emb.nox;
  d_varDeps->at(step) += var.dep;
  d_fixDeps->at(step) += fix.dep;
  d_embDeps->at(step) += emb.dep;
  d_varLucs->at(step) += var.luc;
  d_fixLucs->at(step) += fix.luc;
  d_embLucs->at(step) += emb.luc;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : exportCosts
// ----------------------------------------------------------

boost::tuple
<CostSet,
 CostSet,
 CostSet>
CostRegister::exportCosts
(const int step) const
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  double varFin = d_reset; varFin = d_varFins->at(step);
  double varGhg = d_reset; varGhg = d_varGhgs->at(step);
  double varNox = d_reset; varNox = d_varNoxs->at(step);
  double varDep = d_reset; varDep = d_varDeps->at(step);
  double varLuc = d_reset; varLuc = d_varLucs->at(step);

  double fixFin = d_reset; fixFin = d_fixFins->at(step);
  double fixGhg = d_reset; fixGhg = d_fixGhgs->at(step);
  double fixNox = d_reset; fixNox = d_fixNoxs->at(step);
  double fixDep = d_reset; fixDep = d_fixDeps->at(step);
  double fixLuc = d_reset; fixLuc = d_fixLucs->at(step);

  double embFin = d_reset; embFin = d_embFins->at(step);
  double embGhg = d_reset; embGhg = d_embGhgs->at(step);
  double embNox = d_reset; embNox = d_embNoxs->at(step);
  double embDep = d_reset; embDep = d_embDeps->at(step);
  double embLuc = d_reset; embLuc = d_embLucs->at(step);

  return boost::make_tuple                   // tuple holds copies of its args
    (CostSet(varFin, varGhg, varNox, varDep, varLuc),
     CostSet(fixFin, fixGhg, fixNox, fixDep, fixLuc),
     CostSet(embFin, embGhg, embNox, embDep, embLuc));
```

```
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : consolidateCosts
// ----------------------------------------------------------

// CAUTION: not currently deployed at the time of writing

void
CostRegister::consolidateCosts
(const int                        step,
 const shared_ptr<CostRegister> entity)      // normally a subsidiary entity
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  importCosts(step, entity->exportCosts(step));
}

// ----------------------------------------------------------
//  CLASS           : CostRegisterOverseer
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterOverseer
// ----------------------------------------------------------

CostRegisterOverseer::CostRegisterOverseer
(Record& record) :
  CostRegister(record),
  d_totalFin(record.tieSingle<double>("total-financial")),
  d_totalGhg(record.tieSingle<double>("total-greenhouse")),
  d_totalNox(record.tieSingle<double>("total-nox")),
  d_totalDep(record.tieSingle<double>("total-depletion")),
  d_totalLuc(record.tieSingle<double>("total-landuse"))
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // reset totals
  d_totalFin = d_reset;
  d_totalGhg = d_reset;
  d_totalNox = d_reset;
  d_totalDep = d_reset;
  d_totalLuc = d_reset;

  // tie inherited quantities in the constructor (as opposed to
  // the member initialization list)
  d_varFins = record.tieTimeseries<double>("variable-costs-financial");
  d_fixFins = record.tieTimeseries<double>("fixed-costs-financial");
  d_embFins = record.tieTimeseries<double>("embedded-costs-financial");
  d_varGhgs = record.tieTimeseries<double>("variable-costs-greenhouse");
  d_fixGhgs = record.tieTimeseries<double>("fixed-costs-greenhouse");
  d_embGhgs = record.tieTimeseries<double>("embedded-costs-greenhouse");
  d_varNoxs = record.tieTimeseries<double>("variable-costs-nox");
  d_fixNoxs = record.tieTimeseries<double>("fixed-costs-nox");
  d_embNoxs = record.tieTimeseries<double>("embedded-costs-nox");
  d_varDeps = record.tieTimeseries<double>("variable-costs-depletion");
  d_fixDeps = record.tieTimeseries<double>("fixed-costs-depletion");
  d_embDeps = record.tieTimeseries<double>("embedded-costs-depletion");
  d_varLucs = record.tieTimeseries<double>("variable-costs-landuse");
  d_fixLucs = record.tieTimeseries<double>("fixed-costs-landuse");
  d_embLucs = record.tieTimeseries<double>("embedded-costs-landuse");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterOverseer
// ----------------------------------------------------------

CostRegisterOverseer::~CostRegisterOverseer()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : importCosts
// ----------------------------------------------------------

void
CostRegisterOverseer::importCosts
```

```
(const int step,
 const CostSet&  var,
 const CostSet&  fix,
 const CostSet&  emb)
{
  d_varFins->at(step) += var.fin;
  d_fixFins->at(step) += fix.fin;
  d_embFins->at(step) += emb.fin;
  d_varGhgs->at(step) += var.ghg;
  d_fixGhgs->at(step) += fix.ghg;
  d_embGhgs->at(step) += emb.ghg;
  d_varNoxs->at(step) += var.nox;
  d_fixNoxs->at(step) += fix.nox;
  d_embNoxs->at(step) += emb.nox;
  d_varDeps->at(step) += var.dep;
  d_fixDeps->at(step) += fix.dep;
  d_embDeps->at(step) += emb.dep;
  d_varLucs->at(step) += var.luc;
  d_fixLucs->at(step) += fix.luc;
  d_embLucs->at(step) += emb.luc;

  // CAUTION: 'operator+=' of lower precedence than 'operator+'
  d_totalFin += var.fin + fix.fin + emb.fin;
  d_totalGhg += var.ghg + fix.ghg + emb.ghg;
  d_totalNox += var.nox + fix.nox + emb.nox;
  d_totalDep += var.dep + fix.dep + emb.dep;
  d_totalLuc += var.luc + fix.luc + emb.luc;
}

// -----------------------------------------------------------
//  CLASS           : CostRegisterDomcon
// -----------------------------------------------------------

// -----------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterDomcon
// -----------------------------------------------------------

CostRegisterDomcon::CostRegisterDomcon
(Record& record) :
  CostRegister(record)
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // tie some inherited quantities in the constructor (as opposed
  // to the member initialization list)
  d_varFins = record.tieTimeseries<double>("variable-costs-financial");
  d_fixFins = record.tieTimeseries<double>("fixed-costs-financial");
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterDomcon
// -----------------------------------------------------------

CostRegisterDomcon::~CostRegisterDomcon()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// -----------------------------------------------------------
//  CLASS           : CostRegisterAsop
// -----------------------------------------------------------

// -----------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterAsop
// -----------------------------------------------------------

CostRegisterAsop::CostRegisterAsop
(Record& record) :
  CostRegister(record),
  d_standingFin(record.tieSingle<double>("standing-cost-financial")),
  d_standingGhg(record.tieSingle<double>("standing-cost-greenhouse")),
  d_standingNox(record.tieSingle<double>("standing-cost-nox")),
  d_standingDep(record.tieSingle<double>("standing-cost-depletion")),
  d_standingLuc(record.tieSingle<double>("standing-cost-landuse"))
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // tie some inherited quantities in the constructor (as opposed
  // to the member initialization list)
```

```
  d_varFins = record.tieTimeseries<double>("variable-costs-financial");
  d_fixFins = record.tieTimeseries<double>("fixed-costs-financial");

  // load the cost set variants
  d_standingCosts.fin = d_standingFin;
  d_standingCosts.ghg = d_standingGhg;
  d_standingCosts.nox = d_standingNox;
  d_standingCosts.dep = d_standingDep;
  d_standingCosts.luc = d_standingLuc;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterAsop
// ----------------------------------------------------------

CostRegisterAsop::~CostRegisterAsop()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// ----------------------------------------------------------
//  CLASS           : CostRegisterSRFin
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterSRFin
// ----------------------------------------------------------

CostRegisterSRFin::CostRegisterSRFin
(Record& record) :
  CostRegister(record),
  d_nameplateSize(record.tieSingle<double>("nameplate-capacity")),
  d_dutySpecFin(record.tieSingle<double>("duty-specific-cost-financial")),
  d_sizeSpecFin(record.tieSingle<double>("size-specific-cost-financial")),
  d_standingFin(record.tieSingle<double>("standing-cost-financial"))
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // tie some inherited quantities in the constructor (as opposed
  // to the member initialization list)
  d_varFins = record.tieTimeseries<double>("variable-costs-financial");
  d_fixFins = record.tieTimeseries<double>("fixed-costs-financial");

  // load the cost set variants
  d_dutySpecCosts.fin = d_dutySpecFin;
  d_sizeSpecCosts.fin = d_sizeSpecFin;
  d_standingCosts.fin = d_standingFin;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterSRFin
// ----------------------------------------------------------

CostRegisterSRFin::~CostRegisterSRFin()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// MANIPULATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : updateShortrunCosts (tuple wrapper form)
// ----------------------------------------------------------

void
CostRegisterSRFin::updateShortrunCosts       // wrapper to separated argument version
(const int                              step,
 const boost::tuple<CostSet, CostSet>& shortrun)
{
  // initial reporting
  s_logger_costreg->repx(logga::adhc, "entering member function", "");

  // calculations
  CostRegister::updateShortrunCosts          // CAUTION: note full scope resolution
    (step,
     shortrun.get<0>(),                      // CAUTION: zero-based tuple indexing
     shortrun.get<1>());
}

// ----------------------------------------------------------
```

```
//  MEMBER FUNCTION : updateShortrunCosts
// -------------------------------------------------------

void
CostRegisterSRFin::updateShortrunCosts
(const int    step,
 const double currentDuty)
{
  // initial reporting
  s_logger_costreg->repx(logga::xtra, "entering member function", "");
  std::ostringstream put;
  put << "  step                      : " << step                << "\n"
      << "  current duty              : " << currentDuty         << "\n"
      << "  nameplate capacity        : " << d_nameplateSize     << "\n"
      << "  variable-financial length : " << d_varFins->size()   << "\n";
  s_logger_costreg->putx(logga::adhc, put);

  // calculations and loading
  d_varFins->at(step) = (d_dutySpecFin * currentDuty                    ) * d_interval;
  d_fixFins->at(step) = (d_sizeSpecFin * d_nameplateSize + d_standingFin) * d_interval;
}

// -------------------------------------------------------
//  CLASS           : CostRegisterSRAll
// -------------------------------------------------------

// -------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterSRAll
// -------------------------------------------------------

CostRegisterSRAll::CostRegisterSRAll
(Record& record) :
  CostRegister(record),
  CostRegisterSRFin(record),
  d_dutySpecGhg(record.tieSingle<double>("duty-specific-cost-greenhouse")),
  d_sizeSpecGhg(record.tieSingle<double>("size-specific-cost-greenhouse")),
  d_standingGhg(record.tieSingle<double>("standing-cost-greenhouse")),
  d_dutySpecNox(record.tieSingle<double>("duty-specific-cost-nox")),
  d_sizeSpecNox(record.tieSingle<double>("size-specific-cost-nox")),
  d_standingNox(record.tieSingle<double>("standing-cost-nox")),
  d_dutySpecDep(record.tieSingle<double>("duty-specific-cost-depletion")),
  d_sizeSpecDep(record.tieSingle<double>("size-specific-cost-depletion")),
  d_standingDep(record.tieSingle<double>("standing-cost-depletion")),
  d_dutySpecLuc(record.tieSingle<double>("duty-specific-cost-landuse")),
  d_sizeSpecLuc(record.tieSingle<double>("size-specific-cost-landuse")),
  d_standingLuc(record.tieSingle<double>("standing-cost-landuse"))
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // tie some inherited quantities in the constructor (as opposed
  // to the member initialization list)
  d_varGhgs = record.tieTimeseries<double>("variable-costs-greenhouse");
  d_fixGhgs = record.tieTimeseries<double>("fixed-costs-greenhouse");
  d_varNoxs = record.tieTimeseries<double>("variable-costs-nox");
  d_fixNoxs = record.tieTimeseries<double>("fixed-costs-nox");
  d_varDeps = record.tieTimeseries<double>("variable-costs-depletion");
  d_fixDeps = record.tieTimeseries<double>("fixed-costs-depletion");
  d_varLucs = record.tieTimeseries<double>("variable-costs-landuse");
  d_fixLucs = record.tieTimeseries<double>("fixed-costs-landuse");

// load the cost set variants
  d_dutySpecCosts.ghg = d_dutySpecGhg;
  d_sizeSpecCosts.ghg = d_sizeSpecGhg;
  d_standingCosts.ghg = d_standingGhg;
  d_dutySpecCosts.nox = d_dutySpecNox;
  d_sizeSpecCosts.nox = d_sizeSpecNox;
  d_standingCosts.nox = d_standingNox;
  d_dutySpecCosts.dep = d_dutySpecDep;
  d_sizeSpecCosts.dep = d_sizeSpecDep;
  d_standingCosts.dep = d_standingDep;
  d_dutySpecCosts.luc = d_dutySpecLuc;
  d_sizeSpecCosts.luc = d_sizeSpecLuc;
  d_standingCosts.luc = d_standingLuc;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterSRAll
// -------------------------------------------------------

CostRegisterSRAll::~CostRegisterSRAll()
```

```
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// MANIPULATORS

// -----------------------------------------------------------
//  MEMBER FUNCTION : updateShortrunCosts
// -----------------------------------------------------------

void
CostRegisterSRAll::updateShortrunCosts
(const int    step,
 const double currentDuty)
{
  // initial reporting
  s_logger_costreg->repx(logga::xtra, "entering member function", "");
  std::ostringstream put;
  put << "  step                      : " << step                << "\n"
      << "  current duty              : " << currentDuty         << "\n"
      << "  nameplate capacity        : " << d_nameplateSize     << "\n"
      << "  variable-financial length : " << d_varFins->size()   << "\n";
  s_logger_costreg->putx(logga::adhc, put);

  // calculations and loading
  d_varFins->at(step) = (d_dutySpecFin * currentDuty                    ) * d_interval;
  d_fixFins->at(step) = (d_sizeSpecFin * d_nameplateSize + d_standingFin) * d_interval;
  d_varGhgs->at(step) = (d_dutySpecGhg * currentDuty                    ) * d_interval;
  d_fixGhgs->at(step) = (d_sizeSpecGhg * d_nameplateSize + d_standingGhg) * d_interval;
  d_varNoxs->at(step) = (d_dutySpecNox * currentDuty                    ) * d_interval;
  d_fixNoxs->at(step) = (d_sizeSpecNox * d_nameplateSize + d_standingNox) * d_interval;
  d_varDeps->at(step) = (d_dutySpecDep * currentDuty                    ) * d_interval;
  d_fixDeps->at(step) = (d_sizeSpecDep * d_nameplateSize + d_standingDep) * d_interval;
  d_varLucs->at(step) = (d_dutySpecLuc * currentDuty                    ) * d_interval;
  d_fixLucs->at(step) = (d_sizeSpecLuc * d_nameplateSize + d_standingLuc) * d_interval;
}

// -----------------------------------------------------------
//  CLASS           : CostRegisterEmbFin
// -----------------------------------------------------------

// -----------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterEmbFin
// -----------------------------------------------------------

CostRegisterEmbFin::CostRegisterEmbFin
(Record& record) :
  CostRegister(record),
  d_annualDiscountRate(record.tieSingle<double>("annual-discount-rate-decimal")),
  d_economicLife(record.tieSingle<int>("economic-life")),
  d_capitalInputInitial(record.tieSingle<double>("capex-initial")),
  d_capitalInputTerminal(record.tieSingle<double>("capex-terminal")),
  d_currentAge(record.tieSingle<int>("current-age"))
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // tie some inherited quantities in the constructor (as opposed
  // to the member initialization list)
  d_embFins = record.tieTimeseries<double>("embedded-costs-financial");
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterEmbFin
// -----------------------------------------------------------

CostRegisterEmbFin::~CostRegisterEmbFin()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// MANIPULATORS

// -----------------------------------------------------------
//  MEMBER FUNCTION : updateEmbeddedCosts
// -----------------------------------------------------------

void
CostRegisterEmbFin::updateEmbeddedCosts
(const int step)                            // horizon step
{
```

```
  // initial reporting
  s_logger_costreg->repx(logga::xtra, "entering member function", "");

  // calculations
  // CAUTION: 'xeona::capitalRecovery' return is interval length adjusted
  d_embFins->at(step) = xeona::capitalRecovery(d_annualDiscountRate,
                                               d_economicLife,
                                               d_capitalInputInitial,
                                               d_capitalInputTerminal,
                                               d_currentAge);
}

// ----------------------------------------------------------
//  CLASS           : CostRegisterEmbAll
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CostRegisterEmbAll
// ----------------------------------------------------------

CostRegisterEmbAll::CostRegisterEmbAll
(Record& record) :
  CostRegister(record),
  CostRegisterEmbFin(record),
  d_physicalLife(record.tieSingle<int>("physical-life")),
  d_investGhg(record.tieSingle<double>("investment-greenhouse")),
  d_investNox(record.tieSingle<double>("investment-nox")),
  d_investDep(record.tieSingle<double>("investment-depletion")),
  d_investLuc(record.tieSingle<double>("investment-landuse"))
{
  // initial reporting
  s_logger_costreg->repx(logga::dbug, "constructor call", "");

  // tie some inherited quantities in the constructor (as opposed
  // to the member initialization list)
  d_embGhgs = record.tieTimeseries<double>("embedded-costs-greenhouse");
  d_embNoxs = record.tieTimeseries<double>("embedded-costs-nox");
  d_embDeps = record.tieTimeseries<double>("embedded-costs-depletion");
  d_embLucs = record.tieTimeseries<double>("embedded-costs-landuse");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CostRegisterEmbAll
// ----------------------------------------------------------

CostRegisterEmbAll::~CostRegisterEmbAll()
{
  s_logger_costreg->repx(logga::dbug, "destructor call", "");
}

// MANIPULATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : updateEmbeddedCosts
// ----------------------------------------------------------

void
CostRegisterEmbAll::updateEmbeddedCosts
(const int    step)                          // horizon step
{
  // initial reporting
  s_logger_costreg->repx(logga::xtra, "entering member function", "");

  // calculations
  // CAUTION: 'xeona::capitalRecovery' return is interval length adjusted
  d_embFins->at(step) = xeona::capitalRecovery(d_annualDiscountRate,
                                               d_economicLife,
                                               d_capitalInputInitial,
                                               d_capitalInputTerminal,
                                               d_currentAge);

  // CAUTION: physical life is given in years
  // CAUTION: 'operator*' and 'operator/' associate leftward
  d_embGhgs->at(step) = d_investGhg / d_physicalLife / xeona::secondsPerYear * d_interval;
  d_embNoxs->at(step) = d_investNox / d_physicalLife / xeona::secondsPerYear * d_interval;
  d_embDeps->at(step) = d_investDep / d_physicalLife / xeona::secondsPerYear * d_interval;
  d_embLucs->at(step) = d_investLuc / d_physicalLife / xeona::secondsPerYear * d_interval;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : domcon.cc
//  file-create-date : Thu 07-Aug-2008 20:51 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain controller entity / implementation
//  file-status      : work-in-progress (but nearing completion)
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/domcon.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "domcon.h"           // companion header for this file (place first)

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
#include "../c/recset.h"      // records and fields and also record-sets
#include "../b/teas.h"        // technical asset entity
#include "../b/node.h"        // LMP node entity
#include "../b/junc.h"        // demand split/join junction entity
#include "../b/gate.h"        // gateway entity
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/block.h"       // abstract block entity
#include "../b/asop.h"        // asset operator entity
#include "../a/helpers.h"     // application helper functions

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <deque>              // STL sequence container, double-ended vector
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <utility>            // STL pair, make_pair()

#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting
#include <boost/tuple/tuple.hpp>        // n-tuples, ref(), cref()

//  CODE

// ----------------------------------------------------------
//  FREE FUNCTION   : ::visualize
// ----------------------------------------------------------
//  Description  : create and display solver interface as HTML
//  Role         : currently called within 'DomainController::transolve' under "--yeek 1"
//  Techniques   : optional one pass flag, 'GlpkViz' object
//  Status       : complete
//
//  Typical usage
//
//      ::visualize(d_solver, __func__);
//
//      where '__func__' is a compiler macro
```

```
//
// -------------------------------------------------------

namespace
{
  void
  visualize
  (const shared_ptr<svif::SolverIf> solver,
   const std::string                caller,  // recommended to use __func__
   const int                        step)
  {
    // static variables
    static bool firstPass         = true;
    static logga::spLogger logger = logga::ptrLogStream();  // bind logger

    // initial reporting
    logger->repx(logga::adhc, "entering free function, first pass", firstPass);

    // first pass protection
    if ( firstPass ) firstPass = false;      // toggle the first pass flag
#if 0 // 0 = multiple visualization calls, 1 = one visualization call only
    else             return;
#endif // 0

    // active code
    logger->repx(logga::dbug, "will create and use GlpkViz object", caller);
    std::ostringstream oss;
    oss << solver->getObjectiveSenseStr() << " problem ";   // "maximize" or "minimize"
    if ( ! caller.empty() ) oss << "from function '" << caller << "' at ";
    oss << "step " << step;

    // GlpkViz object
    GlpkViz webbrowse;                       // accept default browser from unit 'common'
    webbrowse(solver,                        // shared pointer (raw pointer also okay)
              oss.str(),                     // comment passed to web page
              4,                             // output precision
              true);                         // use LPX_K_ROUND near-to-zero rounding

    // completion reporting
    logger->repx(logga::adhc, "leaving free function", "");
    // 'webbrowse' will shortly go out of scope

  } // free function 'xeona::visualize'

} // unnamed namespace

// -------------------------------------------------------------
//  FREE FUNCTION   : ::writeProblemInstance
// -------------------------------------------------------------
//  Description  : write out problem (perhaps CPLEX format) and solution (if appropriate)
//  Role         : currently called within 'DomainController::transolve' under "--yeek 15"
//  Techniques   : one pass flag, 'svif::SolverIf::printInfo'
//  Status       : complete
//
//  Typical usage
//
//      ::writeCplexAndSoln(d_solver, __func__);
//
//  Design notes
//
//      Refer to the function 'svif::Solver::printInfo' for
//      information about the exact action.
//
// -------------------------------------------------------------

namespace
{
  void
  writeProblemInstance
  (const shared_ptr<svif::SolverIf> solver,
   const std::string                caller)  // recommended to use __func__
  {
    // static variables
    static bool firstPass         = true;
    static logga::spLogger logger = logga::ptrLogStream();  // bind logger

    // initial reporting
    logger->repx(logga::adhc, "entering free function, first pass", firstPass);

    // first pass protection
    if ( firstPass ) firstPass = false;      // toggle the first pass flag
```

```
    else             return;

    // active code
    logger->repx(logga::dbug, "will write out GLPK problem instance", caller);
    solver->writeInfo();                    // see unit 'siglp' for details
    logger->addDumbBlank();
    logger->repx(logga::adhc, "leaving free function", "");

  } // free function 'xeona::writeProbAndSoln'

} // unnamed namespace

// ------------------------------------------------------------
//  FREE FUNCTION   : ::lowestNotGateway
// ------------------------------------------------------------
//  Description  : traverse buy or selgates and apply function 'func'
//  Role         : called by 'lowest*' functions
//  Techniques   : takes a callback function
//  Status       : complete
//
//  Design notes
//
//      Usage (two statements are recommended)
//
//          funcPtrVoid f = &Gateway::getHash;     // for instance
//          return ::lowestGateway(d_randomBuygates, f, __func__);
//
//  CAUTION: pointer function call syntax
//
//      The following syntax should work for raw pointers, but
//      would NOT with Boost.Smart_ptr smart pointers:
//
//          (gate->*func)()
//
//      The g++ 4.1.2 compiler chokes thus -- see the code proper
//      for a simple work around.
//
//  CAUTION: compiler support for BOOST_FOREACH
//
//      Regarding the local free functions '::lowestNotGateway'
//      and '::lowestGateway' and their traversals.
//
//      The Boost.Foreach macro 'BOOST_FOREACH' may be used to
//      iterate over an expression that returns a sequence by
//      value (that is, an rvalue).  However this can break older
//      compilers -- in which case make 'xeona::reverseSequence'
//      a stand-alone call.  Consult the portability section of
//      Boost.Foreach to see whether your compiler is listed as
//      problematic.
//
//      The foreach loop is never entered if the container is
//      empty, hence all shared_ptr's remain empty.
//
// ------------------------------------------------------------

#define XE_FORWARD 1 // 0 = reverse sequence, 1 = normal order

// The above preprocessor macro applies to the next two free
// functions.  It was added in commit r4563 for experimentation
// purposes and should be removed when the code is stable.

typedef bool (Gateway::* funcPtrVoid)() const;

// The above is a pointer-to-member function type, see Stroustrup
// (1997 p419), Stephens etal (2006 pp539-541), Loudon (2003
// pp26-27).  This typedef makes the subsequent syntax easier.

namespace
{
  shared_ptr<Gateway>
  lowestNotGateway
  (const std::vector<shared_ptr<Gateway> >& gateways,
   funcPtrVoid                              func,
   std::string                              caller)
  {
    // logger
    static logga::spLogger logger = logga::ptrLogStream();

    // active code
    std::string gateId = "(nonexistent)";
    shared_ptr<Gateway> returnme;           // empty smart pointer
```

```
#if (XE_FORWARD == 0)
    BOOST_FOREACH( shared_ptr<Gateway> gate,
                   xeona::reverseSequence(gateways) )
#elif (XE_FORWARD == 1)
    BOOST_FOREACH( shared_ptr<Gateway> gate,
                   gateways )
#endif // XE_FORWARD

      {
        if ( ! ((*gate).*func)() )           // CAUTION: syntax is correct, see above
          {
            returnme = gate;
            gateId   = gate->getIdAndKind();
            break;
          }
      }

    // final reporting
    logger->repx(logga::adhc, "caller: " + caller, gateId);

    // return
    return returnme;
  }

} // unnamed namespace

// ----------------------------------------------------------
//  FREE FUNCTION   : ::lowestGateway
// ----------------------------------------------------------
//  Description  : traverse buy or selgates and apply function 'func'
//  Role         : called by 'lowest*' functions
//  Techniques   : takes a callback function
//  Status       : complete, see comments in ::lowestNotGateway
// ----------------------------------------------------------

namespace
{

  shared_ptr<Gateway>
  lowestGateway
  (const std::vector<shared_ptr<Gateway> >& gateways,
   funcPtrVoid                              func,
   std::string                              caller)
  {
    // logger
    static logga::spLogger logger = logga::ptrLogStream();

    // active code
    std::string gateId = "(nonexistent)";
    shared_ptr<Gateway> returnme;            // empty smart pointer

#if (XE_FORWARD == 0)
    BOOST_FOREACH( shared_ptr<Gateway> gate,
                   xeona::reverseSequence(gateways) )
#elif (XE_FORWARD == 1)
    BOOST_FOREACH( shared_ptr<Gateway> gate,
                   gateways )
#endif // XE_FORWARD

      {
        if ( ((*gate).*func)() )             // CAUTION: syntax is correct, see above
          {
            returnme = gate;
            gateId   = gate->getIdAndKind();
            break;
          }
      }

    // final reporting
    logger->repx(logga::adhc, "caller: " + caller, gateId);

    // return
    return returnme;
  }

} // unnamed namespace

#undef XE_FORWARD

// ----------------------------------------------------------
```

```
//  CLASS           : TestForCostRegister (local class)
// ------------------------------------------------------------
//  Description  : screen for cost entities lacking 'CostRegister' inheritance
//  Role         : used in 'establishDomain' call
//  Techniques   : 'dynamic_pointer_cast'
//  Status       : first-pass complete
// ------------------------------------------------------------

class TestForCostRegister
{
  // DISABLED

private:

  TestForCostRegister(const TestForCostRegister& orig);              // copy constructor
  TestForCostRegister& operator= (const TestForCostRegister& orig);  // copy assign opor

  // CREATORS

public:

  TestForCostRegister() :
    d_failLog()                               // initially empty
  {
    s_logger->repx(logga::xtra, "constructor call", "");
  }

  // MANIPULATORS

  bool
  operator()
  (shared_ptr<Entity> entity)
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    // check first for empty/null shared pointers
    if ( ! entity )
      {
        s_logger->repx(logga::warn, "empty/null pointer supplied", entity);
        return false;
      }

    // report
    s_logger->repx(logga::adhc,
                   "cost register testing cost entity",
                   entity->getIdAndKind());

    // attempt an upcast -- while noting a null pointer indicates failure
    if ( dynamic_pointer_cast<CostRegister>(entity) == 0 )
      {
        d_failLog.push_back(entity->getIdAndKind());
        return false;
      }
    else
      {
        return true;
      }
  }

  // ACCESSORS

  int
  size() const
  {
    return d_failLog.size();
  }

  bool                                        // 'true' means no issues detected
  okay() const
  {
    return d_failLog.size() == 0 ? true : false;
  }

  std::string
  say() const
  {
    std::ostringstream oss;
    BOOST_FOREACH( std::string s, d_failLog )
      {
        oss << "    " << s << "\n";
```

```
      }
    return oss.str();
  }

  // INTERNAL DATA

private:

  std::vector<std::string>    d_failLog;     // vector of collected identifiers
  static logga::spLogger      s_logger;      // shared_ptr to single logger object

};

logga::spLogger TestForCostRegister::s_logger = logga::ptrLogStream();  // bind logger

// ----------------------------------------------------------
//  CLASS           : DomainController
// ----------------------------------------------------------

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : DomainController
// ----------------------------------------------------------

DomainController::DomainController
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  Actor(entityId, record),
  CostRegisterDomcon(record),
  d_initEmployScaling(record.tieSingle<bool>("init-scale-problem")),
  d_initEmployAdvIniBasis(record.tieSingle<bool>("init-use-advanced-initial-basis")),
  d_initSimplexPresolver(record.tieSingle<bool>("init-use-simplex-presolver")),
  d_initIntegerPresolver(record.tieSingle<bool>("init-use-mip-presolver")),
  d_commitment_mode(record.tieSingle<std::string>("commitment-mode")),
  d_commitmentMode(xeona::e_modeNotSpecified),    // will derive from above
  d_ranked_selgates(record.tieSingle<std::string>("ranked-selgates")),
  d_rankedSelgates(),
  d_asset_operators(record.tieSingle<std::string>("asset-operators")),
  d_assetOperators(),                             // empty vector
  d_demand_junctions(record.tieSingle<std::string>("demand-junctions")),
  d_demandJunctions(),
  d_step(-1),                                     // nonsensical value
  d_solver(),                                     // empty object
  d_randomBuygates(),                             // empty vector
  d_tildePushPop(),                               // empty vector
  d_hyphen(false)                                 // initialize
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~DomainController
// ----------------------------------------------------------

DomainController::~DomainController()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : selgateCount
// ----------------------------------------------------------

int                                       // number of selgates (from input data)
DomainController::selgateCount() const
{
  return d_rankedSelgates.size();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : buygateCount
// ----------------------------------------------------------
```

```
int                                             // number of buygates (from DFS traversal)
DomainController::buygateCount() const
{
  return d_randomBuygates.size();
}

// OVERSEER POINTS OF ENTRY

// ----------------------------------------------------------
//  MEMBER FUNCTION : establishDomain
// ----------------------------------------------------------
//  Description  : point of entry for 'establish' calls
//  Role         : beginning of horizon code
//  Techniques   : 'listToVec<>' conversions
//  Status       : first-pass complete
//
//  Design notes
//
//      DFS traversal has run prior to this call.
//
//      Could perhaps move some initialization tasks to 'ctor',
//      but not the 'FullEntity::listToVec' calls.  Otherwise a
//      "hanging association/s found" is reported, followed by an
//      'xeona::xem_data_issue' exception.
//
// ----------------------------------------------------------

void
DomainController::establishDomain()
{
  s_logger->repx(logga::info, "entering member function", "DomainController");

  // ---------------------------------
  //  initializations
  // ---------------------------------

  // call order integrity check
  if ( d_step != -1 )
    {
      s_logger->repx(logga::warn, "d_step not -1 as expected", d_step);
    }

  // commitment mode string to enum mapping
  if      ( d_commitment_mode == "fin"   ) d_commitmentMode = xeona::e_shortrunFin;
  else if ( d_commitment_mode == "ghg"   ) d_commitmentMode = xeona::e_shortrunGhg;
  else if ( d_commitment_mode == "nox"   ) d_commitmentMode = xeona::e_shortrunNox;
  else if ( d_commitment_mode == "dep"   ) d_commitmentMode = xeona::e_shortrunDep;
  else if ( d_commitment_mode == "luc"   ) d_commitmentMode = xeona::e_shortrunLuc;
  else if ( d_commitment_mode == "lmp"   ) d_commitmentMode = xeona::e_auctionLmp;
  else if ( d_commitment_mode == "merit" ) d_commitmentMode = xeona::e_adminMerit;
  else if ( d_commitment_mode == "first" ) d_commitmentMode = xeona::e_adminFirst;
  else
    {
      s_logger->repx(logga::warn, "invalid commitment mode", d_commitment_mode);
      s_logger->repx(logga::info, "commitment mode remains unset", d_commitmentMode);
    }

  // report
  s_logger->repx(logga::dbug, "commitment mode set to", d_commitmentMode);

  // asset operators
  const int asopCnt
    = FullEntity::listToVec<AssetOperator>(d_asset_operators, d_assetOperators);
  s_logger->repx(logga::adhc, "asset operators count", asopCnt);

  // demand junctions
  const int juncCnt
    = FullEntity::listToVec<DemandJunction>(d_demand_junctions, d_demandJunctions);
  s_logger->repx(logga::adhc, "demand junctions count", juncCnt);

  // ranked sell gates
  const int selgateCnt
    = FullEntity::listToVec<Gateway>(d_ranked_selgates, d_rankedSelgates);
  s_logger->repx(logga::adhc, "ranked selgates count", selgateCnt);

  // ---------------------------------
  //  cost register reset
  // ---------------------------------

  CostRegister::resetRegister();            // reset entire register
```

```
  // ---------------------------------
  //  establish sweep (domain)
  // ---------------------------------

  // step thru asset operators in no required order
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      ao->establish();
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      dj->establish();
    }

  // step thru selgates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      sel->establish();
    }

  // step thru buygates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      buy->establish();
    }

  // ---------------------------------
  //  entity authorship integrity
  // ---------------------------------

  // CAUTION: this code requires that all cost entities inherit
  // from 'CostRegister' or one of its sub-classes

  // CAUTION: the following test really only makes sense during
  // development, hence the conditional block

  if ( xeona::releaseStatus == false )
    {

      TestForCostRegister testForCostRegister;    // functor declared and defined above

      // step thru asset operators in no required order
      BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
        {
          testForCostRegister(ao);
          // grab and step thru technical assets in no required order
          std::vector<shared_ptr<TechnicalAsset> > technicalAssets;
          technicalAssets = ao->getTechnicalAssets();
          BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, technicalAssets )
            {
              testForCostRegister(ta);
            }
#if 0 // 0 = LMP nodes do not have cost registers, 1 = otherwise
          // grab and step thru LMP nodes in no required order
          std::vector<shared_ptr<LmpNode> > lmpNodes;
          lmpNodes = ao->getLmpNodes();
          BOOST_FOREACH( shared_ptr<LmpNode> ln, lmpNodes )
            {
              testForCostRegister(ln);
            }
#endif // 0
        }

#if 0 // 0 = gateways do not have cost registers, 1 = otherwise
      // step thru sell gates in rank order, as it happens
      BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
        {
          testForCostRegister(sg);
        }

      // step thru buy gates in arbitrary order
      BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
        {
          testForCostRegister(bg);
        }
#endif // 0
```

```
      // respond appropriately if problems encountered
      if ( testForCostRegister.okay() == false )
        {
          // report
          const int fails = testForCostRegister.size();
          s_logger->repx(logga::warn, "cost entities without registers", fails);
          std::ostringstream put;
          put << "  list of cost entities lacking cost registers (authorship error):"
              << "\n";
          put << testForCostRegister.say();  // contains trailing newline
          s_logger->putx(logga::warn, put);

          // abandon as required
          if ( xeona::nopro == false )       // meaning option '--krazy' not applied
            {
              s_logger->repx(logga::dbug, "will throw xeona::bad_authorship", "");
              throw xeona::bad_authorship(fails);
            }
        }

    } // 'xeona::releaseStatus' conditional

} // member function 'establishDomain'

// ----------------------------------------------------------
//  MEMBER FUNCTION : restructureDomain
// ----------------------------------------------------------
//  Description  : point of entry for 'restructure' calls
//  Role         : structure change code
//  Techniques   : (nothing special)
//  Status       : perhaps complete but not, at the time of writing, called
// ----------------------------------------------------------

void
DomainController::restructureDomain()
{
  // at the time of writing, 'd_commitmentMode' should have been
  // set by 'establishDomain' so complain if not
  if ( d_commitmentMode == xeona::e_modeNotSpecified )
    {
      s_logger->repx(logga::warn, "commitment mode not set properly", d_commitmentMode);
    }

  // ---------------------------------
  //  restructure sweep (domain)
  // ---------------------------------

  // step thru asset operators in no required order
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      ao->restructure(d_commitmentMode);
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      dj->restructure(d_commitmentMode);
    }

  // step thru selgates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      sel->restructure(d_commitmentMode);
    }

  // step thru buygates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      buy->restructure(d_commitmentMode);
    }

} // member function 'restructureDomain'

// ----------------------------------------------------------
//  MEMBER FUNCTION : initializeDomain
// ----------------------------------------------------------
//  Description  : point of entry for 'initialize' calls
//  Role         : beginning of loop code
//  Techniques   : (nothing special)
```

```
//  Status       : first-pass complete
//
//  Design notes
//
//      Function 'TicToc::initialize' is virtual, but is only
//      redefined in a few special cases.
//
// ------------------------------------------------------------

void
DomainController::initializeDomain
(const int step)                            // from 'overseer'
{
  // ---------------------------------
  //  initializations
  // ---------------------------------

  // CAUTION: each domain gets just one solver interface pointer
  // and that this persists for the entire the simulation --
  // notwithstanding the underlying solver interface object is
  // "reset" quite often

  // update interval count
  d_step = step;

  // interpret the current reporting level for transfer to solver
  const int reportLevel                     = s_logger->getReportLevelInt();
  const svif::ReportingLevel svifReportLevel = xeona::logRankToGlpkLevel(reportLevel);

  // process the problem label (the short form is provided for convenience)
  const std::string longform   = "domain-controller-";
  const std::string shortform  = "dc-";
  const std::string identifier = getIdAndKind();
  const int len                = longform.length();
  std::string label            = identifier;
  if ( label.substr(0, len) == longform )
    {
      std::ostringstream put;
      put << "  longform (from the entity identifier specified in model file) : "
          << label << "\n";
      label.replace(0, len, shortform);      // replace call
      put << "  shortform (as revised here)                                   : "
          << label << "\n";
      s_logger->repx(logga::adhc, "revised domain label adopted", "");
      s_logger->putx(logga::adhc, put);
    }
  else
    {
      std::ostringstream put;
      put << "  longform  : " << label << "\n";
      s_logger->repx(logga::adhc, "original domain label retained", "");
      s_logger->putx(logga::adhc, put);
    }

  // reestablish the solver
  d_solver.reset(new svif::SolverIf("", svifReportLevel));
  d_solver->setProblemLabel(label);

  // set close-to-zero rounding (GLPK default is 'false')
  d_solver->setGlpkRounding(xeona::zero);    // 'true' unless '--zero'

  // rework defaults as required
  if ( d_initEmployScaling     ) d_solver->initEmployScaling();
  if ( d_initEmployAdvIniBasis ) d_solver->initEmployAdvBasis();
  if ( d_initSimplexPresolver  ) d_solver->initSimplexPresolver();
  if ( d_initIntegerPresolver  ) d_solver->initIntegerPresolver();

  // CAUTION: there is no need to set the objective function
  // direction and label at this stage -- instead rely on the
  // default settings (namely, 'svif::sense_not_specified' and
  // "" at the time of writing)

  // ---------------------------------
  //  initialize sweep (domain)
  // ---------------------------------

  // CAUTION: the 'TicToc::initialize' calls are polymorphic --
  // moreover, the technical assets are called by their
  // respective operators and not here

  // step thru asset operators in no required order
```

```
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      ao->initialize(d_step, d_solver);
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      dj->initialize(d_step, d_solver);
    }

  // step thru selgates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      sel->SelSide::initialize(d_step, d_solver);
    }

  // step thru buygates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      buy->BuySide::initialize(d_step, d_solver);
    }

} // member function 'initializeDomain'

// ------------------------------------------------------------
//  MEMBER FUNCTION : constrainDomain
// ------------------------------------------------------------

// see r3038 / 08-Jul-2009 for the loop code, but note that
// constraining domains is now part of the CTA process

// ------------------------------------------------------------
//  MEMBER FUNCTION : washupDomain
// ------------------------------------------------------------
//  Description  : point of entry for 'washup' calls
//  Role         : end of loop code
//  Techniques   : (nothing special)
//  Status       : first-pass complete
// ------------------------------------------------------------

void
DomainController::washupDomain()
{
  // ---------------------------------
  //  washup sweep (domain)
  // ---------------------------------

  // step thru asset operators in no required order
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      ao->washup();
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      dj->washup();
    }

  // step thru sell gates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      sel->SelSide::washup();
    }

  // step thru buy gates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      buy->BuySide::washup();
    }

} // member function 'washupDomain'

// ------------------------------------------------------------
//  MEMBER FUNCTION : consolidateDomain
// ------------------------------------------------------------
```

```
//  Description  : point of entry for 'consolidate' calls
//  Role         : end of loop code cost consolidation
//  Techniques   : (nothing special)
//  Status       : first-pass complete
// ----------------------------------------------------------

void
DomainController::consolidateDomain          // dedicated consolidate costs call
(const int step,
 CostSet&  var,
 CostSet&  fix,
 CostSet&  emb)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // CAUTION: code previously run from within 'establishDomain'
  // has already screened for cost entities that lack cost
  // registers (an issue of entity authorship).

  // ---------------------------------
  //  import costs sweep (domain) (consolidateDomain)
  // ---------------------------------

  // step thru asset operators in no required order
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      // grab and step thru technical assets in no required order
      std::vector<shared_ptr<TechnicalAsset> > technicalAssets;
      technicalAssets = ao->getTechnicalAssets();
      BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, technicalAssets )
        {
          // consolidate to asset operator
          ao->importCosts(step, ta->exportCosts(step)); // import employs "add-and-assign"
        }
      // no need to step thru LMP nodes

      // consolidate to domain controller (note the explicit this->)
      this->importCosts(step, ao->exportCosts(step));
    }

  // step thru sell gates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      // consolidate to domain controller (note the explicit this->)
      this->importCosts(step, sel->exportCosts(step)); // import employs "add-and-assign"
    }

  // step thru buy gates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      // consolidate to domain controller (note the explicit this->)
      this->importCosts(step, buy->exportCosts(step)); // import employs "add-and-assign"
    }

  // ---------------------------------
  //  forward data to overseer
  // ---------------------------------

  boost::tie(var, fix, emb) = exportCosts(step);

} // function 'DomainController::consolidateDomain'

// ----------------------------------------------------------
//  MEMBER FUNCTION : concludeDomain
// ----------------------------------------------------------
//  Description  : point of entry for 'conclude' calls
//  Role         : end of horizon code
//  Techniques   : (nothing special)
//  Status       : first-pass complete
// ----------------------------------------------------------

void
DomainController::concludeDomain()
{
  // ---------------------------------
  //  conclude sweep (domain)
  // ---------------------------------
```

```
  // step thru asset operators in no required order
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      ao->conclude();
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      dj->conclude();
    }

  // step thru sell gates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      sel->conclude();
    }

  // step thru buygates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      buy->conclude();
    }

  // ---------------------------------
  //  block stats sweep (domain)
  // ---------------------------------

  // code execution based on current 'logga::Rank' or
  // 'xeona::yeek' values

  const logga::Rank currentRank = s_logger->getReportLevelRank();

  if ( currentRank >= logga::xtra
       ||
       xeona::yeek == 14                    // YEEK 14 CODE (set by '--yeek')
       ||
       xeona::yeek ==  1 )
    {
      std::ostringstream put;

      const int indent         =   4;        // used by 'Block::formatStats' calls
      const int ruleLength     = 126;

      put << "  block summary" << "\n";

      put << std::string(indent, ' ')
          << "domain = "         << getIdAndKind()
          << "   steps = "       << Entity::getHorizonSteps()
          << "   interval = "    << Entity::getHorizonInterval() << " s"  // 's' = second
          << "\n";

      put << Block::formatStatsHeader(indent, ruleLength);  // note trailing newline

      // step thru asset operators in no required order
      BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
        {
          // grab and step thru technical assets in no required order
          std::vector<shared_ptr<TechnicalAsset> > technicalAssets;
          technicalAssets = ao->getTechnicalAssets();
          BOOST_FOREACH( shared_ptr<TechnicalAsset> ta, technicalAssets )
            {
              put << ta->formatStats(indent, "tech");
            }
          // grab and step thru LMP nodes in no required order
          std::vector<shared_ptr<LmpNode> > lmpNodes;
          lmpNodes = ao->getLmpNodes();
          BOOST_FOREACH( shared_ptr<LmpNode> ln, lmpNodes )
            {
              put << ln->formatStats(indent, "node");
            }
        }

      // step thru demand junctions
      BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
        {
          put << dj->formatStats(indent, "junc");
        }
```

```
      // step thru sell gates (only) in rank order, as it happens
      BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
        {
          put << sg->formatStats(indent, "gate");
        }

      put << Block::formatStatsFooter(indent);   // note trailing newline (elsewhere also)

      // CAUTION; note use of 'currentRank'
      s_logger->repx(logga::info, "block duty/size reporting follows", "");
      s_logger->addSmartBlank(currentRank);
      s_logger->putx(currentRank, put);
    }

} // member function 'concludeDomain'

// REGISTER GATEWAYS FUNCTION POINTS OF ENTRY

// the following three 'map*To*' calls are invoked once and once
// only by '::registerGateways'

// ------------------------------------------------------------
//  MEMBER FUNCTION : mapGatesToDomain
// ------------------------------------------------------------

// repackaging of 'd_rankedSelgates'

typedef std::map<shared_ptr<Gateway>,
                 shared_ptr<DomainController> > mapGD_type;

mapGD_type
DomainController::mapGatesToDomain()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "");

  // identify myself
  shared_ptr<DomainController const> meconst = this->retMe<DomainController>(); // [1]
  shared_ptr<DomainController>       me = const_pointer_cast<DomainController>(meconst);
  // [1] CAUTION: template instantiation must be explicit

  // declare buffer
  mapGD_type buffer;

  // process ranked selgates
  FullEntity::listToVec<Gateway>(d_ranked_selgates, d_rankedSelgates);
  BOOST_FOREACH( shared_ptr<Gateway> gate, d_rankedSelgates )
    {
      if ( ! buffer.insert(std::make_pair(gate, me)).second )
        {
          s_logger->repx(logga::warn,
                         "gate insertion unexpectedly failed",
                         gate->getIdAndKind());
        }
    }

  // return
  return buffer;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : mapGatesToBlocks
// ------------------------------------------------------------

// working "up-demand" (often portrayed as leftwards)

typedef std::map<shared_ptr<Gateway>,
                 shared_ptr<Block> > mapGB_type;

mapGB_type
DomainController::mapGatesToBlocks()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "");

  // declare buffer
  mapGB_type buffer;

  // process ranked selgates
  FullEntity::listToVec<Gateway>(d_ranked_selgates, d_rankedSelgates);
  BOOST_FOREACH( shared_ptr<Gateway> gate, d_rankedSelgates )
```

```
    {
      shared_ptr<Block> block = gate->getDemander();   // in { gate, junc, node, teas }
      if ( ! buffer.insert(std::make_pair(gate, block)).second )
        {
          s_logger->repx(logga::warn,
                         "gate insertion unexpectedly failed",
                         gate->getIdAndKind());
        }
    }

  // return
  return buffer;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : mapBlocksToDomain
// ----------------------------------------------------------

// working "down-coordination" (portrayed as downwards)

typedef std::map<shared_ptr<Block>,
                 shared_ptr<DomainController> > mapBD_type;

mapBD_type
DomainController::mapBlocksToDomain()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "");

  // identify myself
  shared_ptr<DomainController const> meconst = this->retMe<DomainController>(); // [1]
  shared_ptr<DomainController>       me = const_pointer_cast<DomainController>(meconst);
  // [1] CAUTION: template instantiation must be explicit

  // declare buffer
  mapBD_type buffer;

  // loop demand junctions
  FullEntity::listToVec<DemandJunction>(d_demand_junctions, d_demandJunctions);
  BOOST_FOREACH( shared_ptr<DemandJunction> junc, d_demandJunctions )
    {
      if ( ! buffer.insert(std::make_pair(junc, me)).second )
        {
          s_logger->repx(logga::warn,
                         "junc insertion unexpectedly failed",
                         junc->getIdAndKind());
        }
    }

  // loop asset operators
  FullEntity::listToVec<AssetOperator>(d_asset_operators, d_assetOperators);
  BOOST_FOREACH( shared_ptr<AssetOperator> asop, d_assetOperators )
    {
      // loop technical assets [1]
      BOOST_FOREACH( shared_ptr<TechnicalAsset> teas, asop->getTechnicalAssets() )
        {
          if ( ! buffer.insert(std::make_pair(teas, me)).second )
            {
              s_logger->repx(logga::warn,
                             "teas insertion unexpectedly failed",
                             teas->getIdAndKind());
            }
        }
      // loop lmp nodes [1]
      BOOST_FOREACH( shared_ptr<LmpNode> node, asop->getLmpNodes() )
        {
          if ( ! buffer.insert(std::make_pair(node, me)).second )
            {
              s_logger->repx(logga::warn,
                             "node insertion unexpectedly failed",
                             node->getIdAndKind());
            }
        }
    }

  // [1] the 'getTechnicalAssets' and 'getLmpNodes' calls now
  // also do 'listToVec' processing

  // return
  return buffer;
}
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : addBuygate
// ----------------------------------------------------------

int                                         // number of buygates following insertion
DomainController::addBuygate
(shared_ptr<Gateway> buygate)
{
  if ( buygate )                            // not empty nor holding null pointer
    {
      d_randomBuygates.push_back(buygate);
      s_logger->repx(logga::adhc, "added gateway", buygate->getIdAndKind());
    }
  else
    {
      s_logger->repx(logga::warn, "attempt to add bad pointer", buygate);
    }
  return d_randomBuygates.size();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retBuygates
// ----------------------------------------------------------

std::vector<shared_ptr<Gateway> >
DomainController::retBuygates()
{
  return d_randomBuygates;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retSelgates
// ----------------------------------------------------------

std::vector<shared_ptr<Gateway> >
DomainController::retSelgates()
{
  return d_rankedSelgates;
}

// DOMAIN FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : getHyphen
//  MEMBER FUNCTION : markHyphen
//  MEMBER FUNCTION : resetHyphen
// ----------------------------------------------------------

bool DomainController::getHyphen() const {                    return d_hyphen; }
bool DomainController::markHyphen()      { d_hyphen = true;  return d_hyphen; }
void DomainController::resetHyphen()     { d_hyphen = false;                  }

// GET GATEWAY FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : lowestNoTildeSel
// ----------------------------------------------------------
//  Description  : 'tilde' is the selgate visited flag
//  Role         : depth first search traversal
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

shared_ptr<Gateway>
DomainController::lowestNoTildeSel()
{
  // additional reporting as appropriate
  // YEEK 27 CODE (set by '--yeek')
  if ( xeona::yeek == 27 || xeona::yeek == 26 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  " << "ranked selgates: tilde statuses" << "\n";
      BOOST_FOREACH( shared_ptr<Gateway> gate, d_rankedSelgates )
        {
          put << "    "
              << std::setw(30) << std::left
              << gate->getIdAndKind() << " "
              << std::boolalpha
              << gate->getTilde() << "\n";
        }
```

```
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // active code
  funcPtrVoid f = &Gateway::getTilde;
  return ::lowestNotGateway(d_rankedSelgates, f, __func__);
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : pushTilde
// -----------------------------------------------------------
//  Description  : push a gateway into 'd_tildePushPop'
//  Role         : depth first search traversal
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//      Prior unit 'f/gatesreg', this function also used to fill
//      'd_randomBuygates'.  That functionality is now contained
//      in 'DomainController::addBuygate' called indirectly from
//      free function 'xeona::registerGates'.
//
// -----------------------------------------------------------

void
DomainController::pushTilde
(shared_ptr<Gateway> gate)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code
  d_tildePushPop.push_back(gate);
  s_logger->repx(logga::adhc, "pushing tilde gateway", gate->getIdAndKind());
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : popTilde
// -----------------------------------------------------------
//  Description  : 'tilde' is the selgate visited flag
//  Role         : depth first search traversal
//  Techniques   : (nothing special)
//  Status       : complete
// -----------------------------------------------------------

shared_ptr<Gateway>                         // return empty on failure
DomainController::popTilde()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code
  shared_ptr<Gateway> gate;                 // empty shared pointer
  if ( d_tildePushPop.empty() )
    {
      s_logger->repx(logga::dbug,
                     "cannot pop tilde gateway (oft okay)",
                     d_tildePushPop.size());
    }
  else
    {
      gate = d_tildePushPop.back();         // grab last element
      d_tildePushPop.pop_back();            // remove last element
      s_logger->repx(logga::adhc,
                     "popping tilde gateway",
                     gate->getIdAndKind());
    }
  return gate;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : lowestNoThetaBuy
// -----------------------------------------------------------
//  Description  : 'theta' is the capacity back-track lock
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// -----------------------------------------------------------
```

```
shared_ptr<Gateway>
DomainController::lowestNoThetaBuy()
{
  funcPtrVoid f = &Gateway::getTheta;
  return ::lowestNotGateway(d_randomBuygates, f, __func__);
}

// ------------------------------------------------------
//  MEMBER FUNCTION : lowestNoStarBuy
// ------------------------------------------------------
//  Description  : 'star' is the transaction forward-track lock
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------

shared_ptr<Gateway>
DomainController::lowestNoStarBuy()
{
  funcPtrVoid f = &Gateway::getStar;
  return ::lowestNotGateway(d_randomBuygates, f, __func__);
}

// ------------------------------------------------------
//  MEMBER FUNCTION : lowestNotTransBuy
// ------------------------------------------------------
//  Description  : 'trans' is transaction
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------

shared_ptr<Gateway>
DomainController::lowestNotTransBuy()
{
  funcPtrVoid f = &Gateway::getTransacted;
  return ::lowestNotGateway(d_randomBuygates, f, __func__);
}

// ------------------------------------------------------
//  MEMBER FUNCTION : lowestNotTransSel
// ------------------------------------------------------
//  Description  : 'trans' is transaction
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------

shared_ptr<Gateway>
DomainController::lowestNotTransSel()
{
  funcPtrVoid f = &Gateway::getTransacted;
  return ::lowestNotGateway(d_rankedSelgates, f, __func__);
}

// ------------------------------------------------------
//  MEMBER FUNCTION : lowestUnderCapBuy
// ------------------------------------------------------
//  Description  : under-capacity means not capset
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------

shared_ptr<Gateway>
DomainController::lowestUnderCapBuy()
{
  funcPtrVoid f = &Gateway::getHash;
  return ::lowestNotGateway(d_randomBuygates, f, __func__);
}

// ------------------------------------------------------
//  MEMBER FUNCTION : lowestUnderCapSel
// ------------------------------------------------------
//  Description  : under-capacity means not capset
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------

shared_ptr<Gateway>
```

```
DomainController::lowestUnderCapSel()
{
  funcPtrVoid f = &Gateway::getHash;
  return ::lowestNotGateway(d_rankedSelgates, f, __func__);
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : lowestSel
// -----------------------------------------------------------
//  Description  : 'sel' means selgate
//  Role         : CTA algorithm
//  Techniques   : (nothing special)
//  Status       : complete
// -----------------------------------------------------------

shared_ptr<Gateway>
DomainController::lowestSel()
{
  funcPtrVoid f = &Gateway::getHash;
  return ::lowestGateway(d_rankedSelgates, f, __func__);
}

// RESET GATEWAY FUNCTIONS

void
DomainController::resetTildeSelgates()
{
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates)
    {
      sg->unmarkTilde();
    }
}

// CAPSET AND TRANSOLVE FUNCTIONS

// -----------------------------------------------------------
//  MEMBER FUNCTION : capset
// -----------------------------------------------------------
//  Description  : set lower and upper domain capacities relative to target gateway
//  Role         : called by a 'CapTransAlg' sub-class (capset and transolve algorithm)
//  Techniques   : minimum and maximum network flow
//  Status       : incomplete (also lower value presumed zero for now) ..see notes in code
//
//  Design notes
//
//      Purpose
//
//          Determine the current technical capacity of a gateway
//          -- which can be materially greater than the offered
//          capacity, an economic process known as withholding.
//
//      Failure to capacitate
//
//          The 'capset' function is not likely to fail but this
//          could conceivably happen in a very badly formed
//          model.  Unless it can be shown that failure is not a
//          possible occurrence, failure needs to be handled in
//          the implementation.
//
//      Capacity mode
//
//          Capacity modes are defined in 'f/ospmodes.h'.  The
//          modes currently supported comprise normal operation,
//          that is, as encoded by the aggregate enumerator:
//
//              xeona::e_normalCapModes
//
//          At this point in development, the
//          'xeona::e_crisisOperation' mode is not supported.
//          This mode indicates that all non-mandatory
//          constraints should be relaxed when generating the
//          domain structure.
//
//      Problem structuring
//
//          The domain constraints are set as per the 'transolve'
//          process.  But in this case, the target selgate
//          contributes unity to the objective function and all
//          other variables contribute zero.  Minimization sets
//          the lower bound and maximization sets the upper
//          bound.
```

```
//
//      Further details
//
//          Refer to my (Robbie Morrison) PhD thesis for a
//          comprehensive description.
//
// ------------------------------------------------------------

bool                                        // return 'false' on failure
DomainController::capset
(const int               step,
 shared_ptr<Gateway>     targetGateway,     // target selgate (meaning sel-side gateway)
 const xeona::DomainMode capacityMode)      // capacity mode indirectly from overseer
{
  // ---------------------------------
  //  preamble
  // ---------------------------------

  // reporting stringstream
  std::ostringstream put;

  // determine aggregate capacity modes
  const xeona::DomainMode aggregateCapacityModes = xeona::e_normalCapModes;

  // undertake outset reporting
  s_logger->repx(logga::dbug, "entering member function", "");
  put << "  function                : " << __func__                        << "\n"
      << "  step (zero-based)       : " << step                            << "\n"
      << "  domain controller       : " << getIdAndKind()                  << "\n"
      << "  target identifier       : " << targetGateway->getIdAndKind()   << "\n"
      << "  given capacity mode     : " << capacityMode                    << "\n"
      << "  aggregate capacity modes : " << aggregateCapacityModes         << "\n";
  s_logger->putx(logga::xtra, put);

  // local quantities
  int targetGol = 0;                        // duty column index for target gateway

  // confirm the capacity mode, noting that argument two (the
  // aggregate) can be less than argument one (the pure)
  if ( ! xeona::isTwoContained(capacityMode, aggregateCapacityModes) )
    {
      std::ostringstream oss;
      oss << "details above if report " << logga::xtra;
      s_logger->repx(logga::warn, "immiscible capacity mode", oss.str());
      s_logger->repx(logga::xtra, "early return", false);
      return false;
    }

  // abandon gracefully if the target gateway is already capacitated
  if ( targetGateway->getHash() == true )    // set hash means "fully-capacitated"
    {
      s_logger->repx(logga::dbug,
                     "target gateway already marked hash",
                     targetGateway->getIdAndKind());
      return true;
    }

  // ---------------------------------
  //  constrain sweep (capset)
  // ---------------------------------

  // reestablish the solver 'd_solver'
  d_solver->resetProblem();                     // nuke for everybody, very neighborly!

  // step thru asset operators
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      xeona::putxId(ao, "DomainController::capset asset operator loop");
      ao->constrain(capacityMode);
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      xeona::putxId(dj, "DomainController::capset demand junction loop");
      dj->constrain(capacityMode);
    }

  // step thru selgates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
```

```
      xeona::putxId(sg, "DomainController::capset sell gate loop");
      // CAUTION: the following action must be done in two statements
      shared_ptr<SelSide> selside = dynamic_pointer_cast<SelSide>(sg);
      const int gol = selside->SelSide::constrain(capacityMode);

      // integrity check
      const std::string selId = sg->getIdAndKind();
      switch ( gol )
        {
        case 0:
          s_logger->repx(logga::warn, "unset duty gol 0", selId);
          break;
        case -1:
          s_logger->repx(logga::warn, "invalid duty gol -1", selId);
          break;
        case -2:
          s_logger->repx(logga::warn, "coded duty gol -2, wrong constrain", selId);
          break;
        default:
          if ( gol < -2 )
            {
              std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
            }
          break;
        }

      // update column index if this particular selgate is the target
      if ( sg == targetGateway )             // based on 'get' result, can also be zero
        {
          s_logger->repx(logga::dbug, "target selgate gol found", gol);
          targetGol = gol;
        }
    }

  // step thru buygates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      xeona::putxId(bg, "DomainController::capset buy gate loop");
      // CAUTION: the following action must be done in two statements
      shared_ptr<BuySide> buyside = dynamic_pointer_cast<BuySide>(bg);
      int gol = buyside->BuySide::constrain(capacityMode);
      static_cast<void>(gol); // temporary to quieten compiler
    }

  // check that we found our target
  if ( targetGol == 0 )
    {
      s_logger->repx(logga::warn, "no target selgate gol", targetGol);
      s_logger->repx(logga::xtra, "early return", false);
      return false;
    }

  // ---------------------------------
  //  replace objective function
  // ---------------------------------

  // grab and store existing objective function and tag
  const std::vector<double> priorObjFunc    = d_solver->getObjective();
  const std::string         priorObjFuncTag = d_solver->getObjectiveTag();

  // create new tag
  const std::string newObjFuncTag
    = "capset"
    + boost::str(boost::format("-%03d") % targetGol);
  s_logger->repx(logga::xtra, "zero-bar-me objective tag", newObjFuncTag);

  // define and install new objective function
  if ( ! d_solver->zeroBarMeObjective(targetGol, 1.0, newObjFuncTag) )
    {
      s_logger->repx(logga::warn, "new objective function load fail", "");
      s_logger->repx(logga::xtra, "early return", false);
      return false;
    }

  // ---------------------------------
  //  flow capacity bounds
  // ---------------------------------

  // first attempt to set the upper limit then the lower limit
  // the following defined array usage is from the 'Boost.Foreach' documentation
```

```
  const svif::ObjectiveSense objectives[] = { svif::maximize, svif::minimize };
  BOOST_FOREACH( svif::ObjectiveSense os, objectives )
    {
      // set objective sense
      d_solver->setObjectiveSense(os, newObjFuncTag);  // set tag again, else ""

      // solve
      d_solver->runSolver();                           // returns void

      // visualize as appropriate
      // YEEK 2 CODE (set by '--yeek')
      if ( xeona::yeek == 2 )
        {
          s_logger->repx(logga::dbug, "entering yeek code", xeona::yeek);
          if ( step == 0 )
            {
              ::visualize(d_solver, __func__, step);  // note caller
            }
        }

      //  write out GLPK problem instance as appropriate
      // YEEK 15 CODE (set by '--yeek')
      if ( xeona::yeek == 15 )
        {
          s_logger->repx(logga::dbug, "entering yeek code", xeona::yeek);
          ::writeProblemInstance(d_solver, __func__); // note caller
        }

      const std::string objLabel = d_solver->getObjectiveLabel();

      // confirm
      if ( ! d_solver->isUsableSoln() )                // returns bool
        {

          s_logger->repx(logga::warn, "solver solution unusable", objLabel);
          s_logger->repx(logga::warn, "solver solution unusable, 1=lo 2=up", os);
          s_logger->repx(logga::xtra, "early return", false);
          return false;
        }

      // recover capacity
      s_logger->repx(logga::adhc, "solver solution usable", "");
      const double capacity = d_solver->getVarValue(targetGol);

      // update data and also report
      switch ( os )
        {
        case svif::maximize:
          s_logger->repx(logga::dbug, "solver solution usable, upper", capacity);
          targetGateway->setUpperCapacity(capacity);
          break;
        case svif::minimize:
          s_logger->repx(logga::dbug, "solver solution usable, lower", capacity);
          targetGateway->setLowerCapacity(capacity);
          break;
        default:                              // compiler needs either default or all enums
          std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
          break;
        }
    }

  // ---------------------------------
  //  finishing up
  // ---------------------------------

  d_solver->renewObjective(priorObjFunc,               // reinstate objective function
                           priorObjFuncTag);

  // ---------------------------------
  //  identify full capacitation
  // ---------------------------------

  // A gateway can have two capacitation states:
  //
  //   * under-capacitation : the innate limits provided by the modeler
  //   * full-capacitation  : the potentially tighter limits provided by this algorithm
  //
  // That is not to say that the full-capacitation limits are
  // immutable -- they can be revised under the hop-relitigate
  // extension to the CTA algorithm
```

```
  const bool priorHash = targetGateway->markHash();    // mark fully-capacitated

  // 'priorHash' should be 'false'
  if ( priorHash == true )
    {
      s_logger->repx(logga::warn, "gateway was fully-capacitated", priorHash);
    }

  // ---------------------------------
  //  return
  // ---------------------------------

  // indicate success
  return true;

} // function 'DomainController::capset'

// ------------------------------------------------------
//  MEMBER FUNCTION : transolve
// ------------------------------------------------------
//  Description  : solve internal flows while honoring selgate obligations
//  Role         : called by the capset and transolve algorithm (CTA)
//  Techniques   : least cost network flow or some variant
//  Status       : incomplete
//
//  Design notes
//
//      Failure to solve
//
//          The 'transolve' function is likely to fail, that is,
//          fail to find a feasible solution.  Such failure needs
//          to be handled in the implementation.
//
//      Commitment modes (aka control strategies)
//
//          Commitment modes are defined in 'f/ospmodes.h'.  The
//          modes currently supported comprise:
//
//              shortrun-fin  : short-run financial cost minimization
//              shortrun-ghg  : short-run ghg contribution minimization
//              shortrun-nox  : short-run nox contribution minimization
//              shortrun-dep  : short-run depletable resource use minimization
//              shortrun-luc  : short-run land use minimization
//
//              auction-lmp   : locational marginal (nodal) pricing auction
//              admin-merit   : prescribed merit order
//              admin-first   : first feasible solution
//
// ------------------------------------------------------

bool                                        // return 'false' on failure
DomainController::transolve
(const int               step,
 const xeona::DomainMode capacityMode)      // capacity mode indirectly from overseer
{
  // ---------------------------------
  //  preamble
  // ---------------------------------

  // reporting stringstream
  std::ostringstream put;

  // determine aggregate capacity modes
  const xeona::DomainMode aggregateCapacityModes = xeona::e_normalCapModes;

  // undertake outset reporting
  s_logger->repx(logga::dbug, "entering member function", "");
  put << "  function                 : " << __func__                << "\n"
      << "  step (zero-based)        : " << step                    << "\n"
      << "  stored commitment mode   : " << d_commitmentMode        << "\n"
      << "  given capacity mode      : " << capacityMode            << "\n"
      << "  aggregate capacity modes : " << aggregateCapacityModes  << "\n";
  s_logger->putx(logga::xtra, put);

  // confirm the capacity mode, noting that argument two (the
  // aggregate) can be less than argument one (the pure)
  if ( ! xeona::isTwoContained(capacityMode, aggregateCapacityModes) )
    {
      std::ostringstream oss;
      oss << "details above if report " << logga::xtra;
      s_logger->repx(logga::warn, "immiscible capacity mode", oss.str());
```

```
      s_logger->repx(logga::xtra, "early return", false);
      return false;
    }

  // any required integrity checks can go here

  // ---------------------------------
  //  constrain sweep (transolve)
  // ---------------------------------

  s_logger->repx(logga::adhc, "domain constrain starts", "");

  // reestablish the solver 'd_solver'
  d_solver->resetProblem();                       // nuke for everybody, very neighborly!

  // step thru asset operators
  BOOST_FOREACH( shared_ptr<AssetOperator> ao, d_assetOperators )
    {
      xeona::putxId(ao, "DomainController::transolve asset operator loop");
      ao->constrain(capacityMode);
    }

  // step thru demand junctions
  BOOST_FOREACH( shared_ptr<DemandJunction> dj, d_demandJunctions )
    {
      xeona::putxId(dj, "DomainController::transolve demand junction loop");
      dj->constrain(capacityMode);
    }

  // step thru selgates in rank order, as it happens
  BOOST_FOREACH( shared_ptr<Gateway> sg, d_rankedSelgates )
    {
      xeona::putxId(sg, "DomainController::transolve sell gate loop");

      // CAUTION: better to cast once when loading
      // 'd_rankedSelgates' and not for each step, this would be
      // faster -- the same applies elsewhere too

      // CAUTION: the following action must be done in two statements
      // this code is buggy: dynamic_pointer_cast<SelSide>(sg)->washup()
      shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(sg);
      sel->SelSide::constrain(capacityMode);
    }

  // step thru buygates in arbitrary order
  BOOST_FOREACH( shared_ptr<Gateway> bg, d_randomBuygates )
    {
      xeona::putxId(bg, "DomainController::transolve buy gate loop");
      shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(bg);
      buy->BuySide::constrain(capacityMode);
    }

  // ---------------------------------
  //  cost-type-based directionality
  // ---------------------------------

  s_logger->repx(logga::adhc, "domain set directionality starts", "");

  // determine the optimization direction using the value of
  // 'd_commitmentMode'.

  svif::ObjectiveSense direction = svif::sense_not_specified;    // initial value
  std::string objFuncTag = "tsolve";

  switch ( d_commitmentMode )
    {
    case xeona::e_shortrunFin:
      direction = svif::minimize;
      objFuncTag += "-srfin";
      break;
    case xeona::e_shortrunGhg:
      direction = svif::minimize;
      objFuncTag += "-srghg";
      break;
    case xeona::e_shortrunNox:
      direction = svif::minimize;
      objFuncTag += "-srnox";
      break;
    case xeona::e_shortrunDep:
      direction = svif::minimize;
      objFuncTag += "-srdep";
```

```
      break;
    case xeona::e_shortrunLuc:
      direction = svif::minimize;
      objFuncTag += "-srluc";
      break;
    case xeona::e_auctionLmp:               // framed as least cost (see end of file)
      direction = svif::minimize;
      objFuncTag += "-nodal";
      break;
    case xeona::e_adminMerit:
      direction = svif::minimize;
      objFuncTag += "-merit";
      break;
    case xeona::e_adminFirst:               // direction not applied in this case
      direction = svif::minimize;
      objFuncTag += "-first";
      break;
    default:
      std::clog << "** coding error 03 in source file " << __FILE__ << std::endl;
      break;
    }

  const std::string oldObjFuncTag = d_solver->getObjectiveTag();
  if ( ! oldObjFuncTag.empty() )
    {
      s_logger->repx(logga::dbug, "object function tag not empty", oldObjFuncTag);
    }

  d_solver->setObjectiveSense(direction, objFuncTag);

  // ---------------------------------
  //  run solver and report
  // ---------------------------------

  s_logger->repx(logga::adhc, "domain run solver starts", "");

  // solve
  d_solver->runSolver();                          // returns 'void'

  // visualize as appropriate
  // YEEK 2 CODE (set by '--yeek')
  if ( xeona::yeek == 2 )
    {
      s_logger->repx(logga::dbug, "entering yeek code", xeona::yeek);
      if ( step == 0 )
        {
          ::visualize(d_solver, __func__, step);  // note caller
        }
    }

  //  write out GLPK problem instance as appropriate
  // YEEK 15 CODE (set by '--yeek')
  if ( xeona::yeek == 15 )
    {
      s_logger->repx(logga::dbug, "entering yeek code", xeona::yeek);
      ::writeProblemInstance(d_solver, __func__); // note caller
    }

  const std::string objLabel = d_solver->getObjectiveLabel();

  // check solution
  if ( ! d_solver->isUsableSoln() )          // returns 'bool'
    {
      // a 'false' return will probably lead to:
      //
      //    application exit : 51 = simulation call encountered
      //    infeasibility (solver fail or problem choke)

      s_logger->repx(logga::warn, "solver solution unusable", objLabel);
      s_logger->repx(logga::warn, "early return", false);
      return false;
    }
  else
    {
      s_logger->repx(logga::dbug, "solver solution usable", objLabel);
    }

  // ---------------------------------
  //  washup sweep (transolve)
  // ---------------------------------
```

```
  // this functionality is now in 'b/overseer.cc:349' within the
  // function 'run' -- so the washups now occur AFTER all the
  // 'transolve' calls (that code could perhaps be moved here if
  // needed)

  // ---------------------------------
  //  finishing up
  // ---------------------------------

  s_logger->repx(logga::adhc, "leaving member function, return", true);

  // indicate success (see above for early returns)
  return true;

} // 'DomainController::transolve'

// ------------------------------------------------------------
//  documentation   : nodal pricing
// ------------------------------------------------------------
//
//  LOCATIONAL MARGINAL PRICING
//
//  Locational marginal pricing (LMP) (also known as "nodal
//  pricing") is a form of wholesale electricity pricing which
//  uses the "reduced cost" (or "slack value") of each nodal flow
//  balance to determine the system unit price for that node.
//
//  LMP applies only to the energy component.  Use of the network
//  itself is treated as a separate issue and transmission
//  pricing methods are applied orthogonally (notwithstanding,
//  the question of a more integrated pricing methodology
//  remains).
//
//  LMP pricing is employed in the PJM (Pennsylvania-New
//  Jersey-Maryland), New York, and New England wholesale
//  electricity markets in the USA, in New Zealand, and
//  elsewhere.  Much of the underpinning theory is based on work
//  by William Hogan, Harvard University.  LMP markets have been
//  operating since the mid-1990s, with mixed success in the case
//  of New Zealand.
//
//  The reduced cost for each nodal flow balance is calculated
//  using a direct current (DC) power flow model and a linear
//  program (LP).  The total cost of supply and demand is
//  minimized using the bid unit prices.  The resultant reduced
//  cost for each nodal flow balance sets the system unit price
//  (or "nodal" price) for that node.  If the bids are
//  cost-reflective, the nodal price represents the opportunity
//  cost of export.  However, bids need not be cost-reflective
//  and may indeed be highly strategic.
//
//  If a generator (or rather a bid) is:
//
//    * partially dispatched : nodal price = offer price
//    * fully dispatched     : nodal price > than offer price
//    * not dispatched       : nodal price < than offer price
//
//  In addition, system constraints can result in a lack of
//  competition.  Provoking constraints for the purpose of
//  profiteering is known as constraint gaming.  One form of
//  gaming tactic is to fake an unscheduled outage (used by Enron
//  during the Californian power crisis in 2000).
//
//  The market operator normally also acts as the system banker.
//  Under LMP pricing, a surplus normally accrues.  This then
//  needs to be redistributed, and is often handed to the
//  generators.  In some jurisdictions, this surplus is assigned
//  to the transmission operator -- a practice not generally
//  favored as it offers the transmission operator a perverse
//  incentive to increase grid losses.  Cash flows are not
//  modeled here but can be easily calculated by hand.
//
//  The algorithm implemented here employs full calculation.
//  Faster algorithms exist, including the reference node method.
//
//  SOME IMPLEMENTATION ISSUES
//
//  Transmission losses are quadratic on power flow, which means
//  that the relative losses (defined as one - efficiency) need
//  to be stepwise discretized.  Generator bids (and also
//  demander bids) likewise need to be stepwise.
```

```
//
//  The LP problem formulation adopted here assumes that power
//  flows are non-negative.  Transmission lines are naturally
//  bi-directional and hence the 'line' component
//  characterization requires equations for both forward (fwd)
//  and back (bak) flow.

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : entity.cc
//  file-create-date : Fri 15-Jun-2007 08:52 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : entity base class plus lazy linking / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/entity.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "entity.h"           // companion header for this file (place first)

#ifndef _XUTEST  // this greatly simplifies unit testing dependencies
# include "../b/asop.h"       // asset operator entity
# include "../b/domcon.h"     // domain controller entity
# include "../b/gate.h"       // gateway entity
# include "../b/junc.h"       // demand split/join junction entity
# include "../b/node.h"       // LMP node entity
# include "../b/overseer.h"   // top-level overseer entity (singleton)
# include "../b/teas.h"       // technical asset entity
#else
  class AssetOperator    { }; // hollow definition
  class DemandJunction   { }; // hollow definition
  class DomainController { }; // hollow definition
  class Gateway          { }; // hollow definition
  class Overseer         { }; // hollow definition
  class TechnicalAsset   { }; // hollow definition
  class LmpNode          { }; // hollow definition
#endif // _XUTEST

#include "../c/factory.h"     // entity factory
#include "../c/recset.h"      // record set support
#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <iterator>           // STL additional iterators, std::distance()
#include <sstream>            // string-streams
#include <typeinfo>           // run-time type information (RTTI)

#include <cctype>             // C-style char classification, case conversion

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ----------------------------------------------------------
//  CLASS           : Entity
// ----------------------------------------------------------
//  Description  : abstract base class for all entities
//  Role         : provide basic housekeeping and class-wide support
//  Techniques   : static censuses, shared pointers, RTTI (run-time type identification)
```

```
//  Status       : working but incomplete
//
//  Design notes
//
//      See header
//
//          Some introductory comments are also given in the
//          header file.
//
//      Raw census
//
//          A list (and not vector) is used for 's_censusRaw' for
//          ease of element removal.  In addition, one of the
//          Boost.Ptr_containers was considered.  But no
//          compelling advantages materialized, given the simple
//          usage here -- no polymorphic behavior, no null
//          pointers, and no small object efficiency
//          considerations.  Boost.Any objects were also
//          considered but again no clear advantages were likely
//          to accrue.
//
//          Unfortunately smart 'this' pointers cannot be used in
//          constructors (as documented in the code proper).
//
//          Stephens etal (2006 pp294-295) provide similar code
//          to that used here except that their destructor
//          searches for its own 'this' entry -- here the
//          'remove(value)' member function is used.
//
//      shared_from_this()
//
//          During development, this class variously used
//          'shared_from_this' and explicitly processed
//          'shared_ptr's from the entity factory.  Currently,
//          the shared pointers derive from calls to the
//          '::createType' function template.
//
//          The Boost documentation states: "the
//          shared_from_this() member function (actually there
//          are two, depending on constness) REQUIRES: .. there
//          must exist at least one shared_ptr instance .. that
//          owns .. [the underlying object]"
//
//          see: file:///usr/share/doc/libboost-doc/HTML/libs/
//          smart_ptr/enable_shared_from_this.html
//
//      Censuses
//
//          The primary census container 's_censusFull' contains
//          the canonical shared pointers.  The container
//          's_censusLink' contains "link" entity shared pointers
//          -- which will, in due course, be redirected to "full"
//          entities as a result of the 'linkAll' call.
//
//          The raw pointer census 's_censusRaw' is provided only
//          for development purposes.
//
//      Stream insertion
//
//          The 'xeona' object factory returns a (std::tr1:: or
//          boost::) shared_ptr<Entity> and it would be
//          convenient to correctly stream sub-entities without
//          additional code:
//
//              std::ostream << *subent;
//
//          The usual free function granted friendship approach
//          (see, for example, Stephens etal 2006 pp363-366) does
//          not allow for polymorphic behavior.
//
//          The helper variant used here is from Dattatri (2002
//          pp356-357 and surrounding text) with my contribution
//          being the addition of 'virtual' to facilitate
//          polymorphic behavior.  Note that most stream
//          insertions will call the free (non-member) overloaded
//          stream insertion operator:
//
//              ostream& operator<< (ostream& os, const Entity& en)
//
//          That said, sub-entities are also given there own
//          dedicated overloaded stream insertion operators.
```

```
//
//          Finally, Sutter and Alexandrescu (2005 p79) cover
//          this topic in a more general sense, stating: "add a
//          virtual member function to provide the virtual
//          behavior, and implement the nonmember [free function]
//          in terms of that [member function]".
//
// ----------------------------------------------------------

//  STATIC DEFINITIONS - with and without explicit initialization

int Entity::s_horizonSteps            = -1;  // negative number means value not yet set
int Entity::s_horizonInterval         = -1;  // negative number means value not yet set
int Entity::s_horizonStartHour        = -1;  // negative number means value not yet set
int Entity::s_horizonStartDay         = -1;  // negative number means value not yet set
xeona::Hemisphere Entity::s_horizonHemisphere = xeona::e_notSet;

int Entity::s_linkAllCallCount = 0;

std::list<const Entity*>          Entity::s_censusRaw;
std::vector<shared_ptr<Entity> >  Entity::s_censusFull;
std::vector<assign_ptr<Entity> >  Entity::s_censusLink;
std::vector<std::string>          Entity::s_identsLink;

std::vector<std::string> Entity::s_entityTypesWant;
std::vector<std::string> Entity::s_entityTypesGot;
std::vector<std::string> Entity::s_entityTypesDiff;

LinkLogger      Entity::s_linkLogger;
logga::spLogger Entity::s_logger = logga::ptrLogStream();

Record      Entity::s_recordStandIn;         // zero-argument constructed 'Record'
std::string Entity::s_stringEmpty = "";
double      Entity::s_doubleZero  = 0.0;
int         Entity::s_intZero     = 0;
bool        Entity::s_boolFalse   = false;

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : Entity
// ----------------------------------------------------------
//  Description  : constructor
//  Status       : complete
// ----------------------------------------------------------

Entity::Entity
(const std::string entityId,                // enforced unique identifier
 Record&           record) :                // CAUTION: pass-by-ref necessary
   d_identifier(entityId),
   d_record(record),
   d_typeStr()
{
  s_logger->repx(logga::dbug, "constructor call", d_identifier);

  const int idLength  = d_identifier.length();
  const int maxLength = 32;
  if (  idLength > maxLength )
    {
      std::ostringstream oss1;
      oss1 << "entity identifier exceeds " << maxLength << " chars";
      std::ostringstream oss2;
      oss2 << d_identifier << " " << idLength;
      s_logger->repx(logga::rankJumpy, oss1.str(), oss2.str());
    }

  // CAUTION: Becker (2007 p58 footnote 5) says NOT to use
  // shared_from_this() -- which provides support for shared
  // pointer 'this' pointers -- in constructor definitions.

  Entity::s_censusRaw.push_back(this);       // insert copy of pointer in raw census
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Entity
// ----------------------------------------------------------
//  Description  : destructor
//  Techniques   : virtual, protected
//  Status       : complete
// ----------------------------------------------------------
```

```
Entity::~Entity()
{
  s_logger->repx(logga::xtra, "destructor call", d_identifier);

  // Remove copy of 'this' pointer from the instance census
  // 's_censusRaw'.  Aggregate (potentially multiple identical)
  // element removal is straightforward with with a list but not
  // a vector -- see Stephens etal (2006 p230) for more
  // information (whilst noting that 'this' should be, by
  // definition, unique!).
  //
  // As a digression, Lischner (2003 p353) offers alternative
  // code for vectors, but data.erase() (rather than his typo
  // std::erase) is correct as far as I can tell.  Finally,
  // Josuttis (1999 p378) discusses the issue.

  Entity::s_censusRaw.remove(this);     // 'remove' is 'std::list' but not 'std::vector'
}

// ------------------------------------------------------
//  MEMBER FUNCTION : factoryInitialize
// ------------------------------------------------------
//  Description  : post-construction initialization by factory call
//  Role         : to support the set-up of entities
//  Caller       : one of the 'EntityFactory' '::createType<>' functions
//  Status       : complete
// ------------------------------------------------------

void
Entity::factoryInitialize()
{
  s_logger->repx(logga::warn, "Entity instance initialize", "base class form");
}

// ACCESSORS

// ------------------------------------------------------
//  MEMBER FUNCTION : getIdentifier
// ------------------------------------------------------
//  Description  : returns unique entity identifier
//  Role         : accessor
//  Status       : complete
// ------------------------------------------------------

const std::string
Entity::getIdentifier() const
{
  return d_identifier;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : getIdAndKind
// ------------------------------------------------------
//  Description  : returns unique entity identifier and if "link" entity
//  Role         : accessor
//  Status       : complete
// ------------------------------------------------------

const std::string
Entity::getIdAndKind                          // returns d_identifier and if link entity
(const std::string& auxMsg) const             // optional auxillary message, note default
{
  std::string buffer = d_identifier;
  if ( getTypeStr().empty() ) buffer += " (link)";   // proxy for link entity
  else                        buffer += "";          // could also be " (full)"
  if ( ! auxMsg.empty() )     buffer += " [" + auxMsg + "]";
  return buffer;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : getTypeStr
// ------------------------------------------------------
//  Description  : returns RTTI (run-time type information) type string
//  Role         : accessor
//  Status       : complete
// ------------------------------------------------------

const std::string
Entity::getTypeStr() const
{
  return d_typeStr;
```

```
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getUseCount
// -------------------------------------------------------
//  Description  : return number of shared pointer in existence including the census copy
//  Role         : accessor
//  On fail      : returns 0
//  Techniques   : 'shared_ptr::use_count'
//  Status       : complete
// -------------------------------------------------------

const int
Entity::getUseCount() const                  // includes copy in census
{
  const shared_ptr<Entity> me = retConstMe();
  if ( ! me )                                // 'retConstSharedPtr' returns empty on fail
    {
      s_logger->repx(logga::warn, "could not obtain my shared pointer", d_identifier);
      return 0;
    }
  int useCount = me.use_count();
  return --useCount;                         // minus one for 'me'
}                                            // 'me' goes out of scope

// -------------------------------------------------------
//  MEMBER FUNCTION : report
// -------------------------------------------------------
//  Description  : prepare a report
//  Role         : debugging
//  Caller       : various
//  Techniques   : 'typeid' 'getIdentifier'
//  Status       : complete
// -------------------------------------------------------

const std::string
Entity::report
(const std::string msg) const                // note default
{
  s_logger->repx(logga::adhc, "entering member function", msg);

  const std::string etype = xeona::demangle(typeid(*this).name());
  std::ostringstream oss;
  oss << "  entity report (this-based):"                    << "\n"
      << "    caller message : " << msg                     << "\n"
      << "    address        : " << this                    << "\n"
      << "    id             : " << this->getIdentifier()   << "\n"
      << "    type construct : " << this->getTypeStr()      << "\n"
      << "    type current   : " << etype                   << "\n";
  return oss.str();
}

// STATIC MANIPULATORS -- for the "lazy" linking process

// -------------------------------------------------------
//  MEMBER FUNCTION : linkAll (static)
// -------------------------------------------------------
//  Description  : traverses and remaps referenced "link" pointers to "full" entities
//  Role         : final step in the entity linking process
//  Robustness   : repeat calls should not cause a problems
//  Techniques   : references for shared pointers (for the link holding entity)
//  Status       : complete
//
//  Design notes
//
//      An entity holding a link to another entity INITIALLY does
//      so by reference to a shared pointer containing a "link"
//      entity with the required identifier string.  The "link"
//      entities are duly placed in a "link" census.
//
//      This function simply remaps the "link" shared pointer to
//      its proper "full" entity.
//
//      The "link" census 'std::vector' needs to remain intact
//      after this function call because its shared pointer
//      contents are still required by the requiring-links
//      entities. (I had originally thought that 'clearing' the
//      "link" census after this remapping would be correct).
//
//      In any case, the "link" entities are duly deleted and
//      release their memory (not that this would amount to
```

```
//      much).
//
//      Regarding repeat calls, this function should be
//      dynamic-safe in the sense that new entities could be
//      constructed and linked -- that said, entity removal is
//      more problematic and is unlikely to be supported in the
//      near future.
//
// -----------------------------------------------------

bool
Entity::linkAll()
{
  // initial reporting, also update number of calls
  ++s_linkAllCallCount;
  s_logger->repx(logga::dbug, "entering member function, call cnt", s_linkAllCallCount);

  // reset the link logger
  s_linkLogger.reset();                       // class 'LinkLogger'

  // traverse link census
  BOOST_FOREACH( assign_ptr<Entity> link, s_censusLink )
    {
      // do the linking, then log success or count failures
      const bool okay = link->polylink(link);
      s_linkLogger.insert(link, okay);
    }

  // grab some counts
  const int links = s_linkLogger.getLogCount();
  const int fails = s_linkLogger.getNullCount();

  // report the outcome
  s_logger->repx(logga::info, "link (informational) success count", links);
  if ( fails > 0 ) s_logger->repx(logga::warn, "link fail count", fails);
  else             s_logger->repx(logga::xtra, "link fail count", fails);
  std::ostringstream put;
  put << s_linkLogger.recover();
  s_logger->putx(logga::dbug, put);

  // return status
  if ( fails > 0 ) return false;
  else             return true;
}

// STATIC ACCESSORS

// -----------------------------------------------------
//  MEMBER FUNCTION : getFullPopn (static)
// -----------------------------------------------------
//  Description  : returns current population
//  Role         : static accessor for test purposes
//  Techniques   : 'std::vector::size' member function
//  Status       : complete
// -----------------------------------------------------

int
Entity::getFullPopn()
{
  return s_censusFull.size();                // just the genuine entities
}

// -----------------------------------------------------
//  MEMBER FUNCTION : getHorizonSteps (static)
// -----------------------------------------------------
//  Description  : returns current simulation horizon steps
//  Role         : static accessor
//  On fail      : returns 0
//  Status       : complete
// -----------------------------------------------------

const int
Entity::getHorizonSteps()
{
  if ( s_horizonSteps < 0 )
    {
      s_logger->repx(logga::warn, "horizon steps not initialized", s_horizonSteps);
      return 0;
    }
  return s_horizonSteps;
}
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : getHorizonInterval (static)
// ----------------------------------------------------------
//  Description  : returns current simulation horizon interval length
//  Role         : static accessor
//  On fail      : returns 0
//  Status       : complete
// ----------------------------------------------------------

const int
Entity::getHorizonInterval()
{
  if ( s_horizonInterval < 0 )
    {
      s_logger->repx(logga::warn, "horizon interval not initialized", s_horizonInterval);
      return 0;
    }
  return s_horizonInterval;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getHorizonStartHour (static)
// ----------------------------------------------------------
//  Description  : returns current simulation horizon start hour
//  Role         : static accessor
//  On fail      : returns 0
//  Status       : complete
// ----------------------------------------------------------

const int
Entity::getHorizonStartHour()
{
  if ( s_horizonStartHour < 0 )
    {
      s_logger->repx(logga::warn, "horizon start hour uninitialized", s_horizonStartHour);
      return 0;
    }
  return s_horizonStartHour;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getHorizonStartDay (static)
// ----------------------------------------------------------
//  Description  : returns current simulation horizon start day
//  Role         : static accessor
//  On fail      : returns 0
//  Status       : complete
// ----------------------------------------------------------

const int
Entity::getHorizonStartDay()
{
  if ( s_horizonStartDay < 0 )
    {
      s_logger->repx(logga::warn, "horizon start day not initialized", s_horizonStartDay);
      return 0;
    }
  return s_horizonStartDay;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getHorizonHemisphere (static)
// ----------------------------------------------------------
//  Description  : returns current simulation horizon hemisphere
//  Role         : static accessor
//  On fail      : returns xeona::e_notSet (also 0)
//  Status       : complete
// ----------------------------------------------------------

const xeona::Hemisphere
Entity::getHorizonHemisphere()
{
  if ( s_horizonHemisphere < xeona::e_notSet )
    {
      s_logger->repx(logga::warn,"horizon hemisphere uninitialized", s_horizonHemisphere);
      return xeona::e_notSet;
    }
  return s_horizonHemisphere;
}
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : confirmIdentifier (static)
// ----------------------------------------------------------
//  Description  : hunts for sought identifier in "full" census
//  Role         : used by 'xeona::simulate'
//  On fail      : returns 'false'
//  Status       : complete
// ----------------------------------------------------------

const bool                                  // 'true' if in "full" census, else 'false'
Entity::confirmIdentifier
(const std::string& soughtId)
{
  BOOST_FOREACH( shared_ptr<Entity> s, s_censusFull )
    {
#ifdef _XUTEST
      std::ostringstream put;
      put << "    id : " << s->d_identifier << "\n";
      s_logger->putx(logga::dbug, put);
#endif // _XUTEST
      if ( s->d_identifier == soughtId )
        return true;
    }
  return false;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retOverseer (static)
// ----------------------------------------------------------

shared_ptr<Overseer>
Entity::retOverseer()
{
  // 'xeona::overseer' is defined in 'common.cc'
  return dynamic_pointer_cast<Overseer>(retSharedPtr(xeona::overseer));
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retDomains (static)
// ----------------------------------------------------------

std::vector<shared_ptr<DomainController> >
Entity::retDomains()
{
  std::vector<shared_ptr<DomainController> > buffer;
  BOOST_FOREACH( shared_ptr<Entity> entity, s_censusFull )
    {
      // attempt a downcast
      shared_ptr<DomainController> dom = dynamic_pointer_cast<DomainController>(entity);
      if ( dom != 0 )
        {
          buffer.push_back(dom); // load the downcast version
        }
    }
  return buffer;                            // can be empty
}

// STATIC ACCESSORS FOR TEST PURPOSES

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLinkPopn (static)
// ----------------------------------------------------------
//  Description  : returns current "link" entity population
//  Role         : static accessor for test purposes
//  Techniques   : 'std::vector::size' member function
//  Status       : complete
// ----------------------------------------------------------

int
Entity::getLinkPopn()
{
  return s_censusLink.size();               // the number of (formed or unformed) links
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getRawPopn (static)
// ----------------------------------------------------------
//  Description  : returns current population
//  Role         : static accessor for test purposes
//  Techniques   : 'std::list::size' member function
//  Status       : complete
```

```
// ------------------------------------------------------

int
Entity::getRawPopn()
{
  return s_censusRaw.size();                 // "full" and "link"(if any) entities
}

// ------------------------------------------------------
//  MEMBER FUNCTION : getCensusRaw (static)
// ------------------------------------------------------
//  Description  : returns 'const' reference to 's_censusRaw'
//  Role         : static accessor for test purposes
//  Status       : complete
//
//  CAUTION: on returning references
//
//      See Dattatri (2002 pp109-112) about returning references
//      from functions -- in particular, be careful that the
//      referenced object outlasts the reference holder.
//
// ------------------------------------------------------

const std::list<const Entity*>&              // return a reference to a 'const' container
Entity::getCensusRaw()
{
  s_logger->repx(logga::dbug, "raw census reference, size", s_censusRaw.size());
  return s_censusRaw;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : traverseFullPopulation (static)
// ------------------------------------------------------
//  Description  : returns string with entity identifiers
//  Role         : static accessor for test purposes
//  Techniques   : string-stream
//  Status       : complete
// ------------------------------------------------------

std::string                                  // returns string for test purposes
Entity::traverseFullPopulation()
{
  std::ostringstream ss;
  ss << "traversal : |";
  BOOST_FOREACH( shared_ptr<Entity> s, s_censusFull )
    {
      ss << " " << s->d_identifier << " |";
    }
  ss << "\n";
  return ss.str();
}

// STATIC MANIPULATORS

// ------------------------------------------------------
//  MEMBER FUNCTION : setHorizonSteps (static)
// ------------------------------------------------------
//  Description  : sets simulation horizon steps
//  Role         : static manipulator
//  Techniques   : static data (to count calls), string-stream
//  Status       : complete
// ------------------------------------------------------

bool
Entity::setHorizonSteps
(const int steps)
{
  bool ret = true;

  static bool firstCall = true;
  if ( firstCall == false )
    {
      s_logger->repx(logga::warn, "illegal reset, firstCall", firstCall);
      ret = false;
    }
  firstCall = false;

  if ( s_horizonSteps != -1 )
    {
      s_logger->repx(logga::warn, "illegal reset, s_horizonSteps", s_horizonSteps);
      ret = false;
```

```
    }

  const int was  = s_horizonSteps;           // briefly capture for reporting
  s_horizonSteps = steps;                    // reset static variable

  std::ostringstream oss;
  oss << was << " > " << s_horizonSteps;
  s_logger->repx(logga::xtra, "set s_horizonSteps, was > now", oss.str());

  return ret;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : addWant (static)
// ----------------------------------------------------------
//  Description  : add a Want
//  Role         : add an entity to the mandatory list
//  Techniques   : pass-by-ref
//  Status       : complete
// ----------------------------------------------------------

void
Entity::addWant
(const std::string& entityType)
{
  s_logger->repx(logga::dbug, "adding wanted entity type", entityType);
  s_entityTypesWant.push_back(entityType);   // normal STL copy in
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : checkWants (static)
// ----------------------------------------------------------
//  Description  : process Wants and Gots, pass out Diffs list
//  Role         : confirm that all mandatory entities exist
//  Techniques   : pass-by-ref argument, calls 'checkEntityTypesLists'
//  Status       : complete
// ----------------------------------------------------------

void
Entity::checkWants
(std::vector<std::string>& buffer)
{
  checkEntityTypesLists();                               // static utility function
  std::vector<std::string> local(s_entityTypesDiff);   // make copy
  buffer = local;                                        // bind copy
}

// UTILITY FUNCTIONS -- non-public

// ----------------------------------------------------------
//  MEMBER FUNCTION : isUniqueIdentifierFull
// ----------------------------------------------------------
//  Description  : check given "full" sub-entity identifier is unique
//  Role         : integrity checking during construction
//  On success   : return 'true'
//  Techniques   : simple string match
//  Status       : complete
// ----------------------------------------------------------

bool                                         // 'true' indicates unique
Entity::isUniqueIdentifierFull()
{
  BOOST_FOREACH( shared_ptr<Entity> s, s_censusFull )
    {
      if ( s->d_identifier == d_identifier ) // simple string match
        {
          s_logger->repx(logga::warn, "non-unique id, d_identifier", d_identifier);
          return false;
        }
    }
  return true;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retConstMe
// ----------------------------------------------------------

const shared_ptr<Entity>
Entity::retConstMe() const                   // wrapper call to 'getConstSharedPtr'
{
  return retConstSharedPtr(d_identifier);
```

```
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retConstSharedPtr (static)
// ----------------------------------------------------------
//  Description  : effectively returns 'const' shared pointer to 'this'
//  Role         : accessor
//  On fail      : returns empty shared pointer
//  Techniques   : wrapper call to 'getNonConstSharedPtr'
//  Status       : complete
// ----------------------------------------------------------

const shared_ptr<Entity>
Entity::retConstSharedPtr
(const std::string& soughtId)
{
  return retSharedPtr(soughtId);             // 'const' added by return type
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : retSharedPtr (static)
// ----------------------------------------------------------
//  Description  : effectively returns non-'const' shared pointer to 'this'
//  Role         : accessor and potential manipulator
//  On fail      : returns empty shared pointer
//  Techniques   : census search
//  Status       : complete
// ----------------------------------------------------------

shared_ptr<Entity>
Entity::retSharedPtr
(const std::string& soughtId)
{
  s_logger->repx(logga::xtra, "entering member function, sought id", soughtId);

  // SEARCHING: generally speaking, use of the 'std::find'
  // function template from <algorithm> would be considered
  // better style, but that approach also requires a custom
  // comparison functor (or free function) -- the following code
  // is therefore fine for our purposes (similar comments could
  // apply elsewhere in this unit)

  BOOST_FOREACH( shared_ptr<Entity> s, s_censusFull )
    {
      if ( s->d_identifier == soughtId )     // simple string match
        {
          return s;
        }
    } // BOOST_FOREACH

  s_logger->repx(logga::info, "returning empty, no match for id", soughtId);
  return shared_ptr<Entity>::shared_ptr(); // else return empty shared pointer
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : checkEntityTypesLists (static)
// ----------------------------------------------------------
//  Description  : performs a set-theoretic Want / Got, returns 'true' for null set
//  Role         : checking for mandatory entities
//  Techniques   : 'std::set_difference' from <algorithm>
//  Status       : complete
// ----------------------------------------------------------

bool
Entity::checkEntityTypesLists()
{
  s_logger->repx(logga::dbug, "entering static member function", "");

  // create Got list
  BOOST_FOREACH( shared_ptr<Entity> s, s_censusFull )
    {
      const std::string classStr = (s->d_record).locateClass();
      s_logger->repx(logga::xtra, "registering entity type on got list", classStr);
      s_entityTypesGot.push_back(classStr);      // register on got list
    }

  // some convenient typedefs
  typedef std::vector<std::string> ets_list;                    // see class declaration
  typedef ets_list::iterator       ets_list_iter;

  // sort the input containers
```

```
  std::sort(s_entityTypesWant.begin(),                              // [1]
            s_entityTypesWant.end());

  std::sort(s_entityTypesGot.begin(),
            s_entityTypesGot.end());

  // "remove" consecutive duplicates
  ets_list_iter entityTypesWant_end =                               // new logical end
    std::unique(s_entityTypesWant.begin(),                          // [2]
                s_entityTypesWant.end());

  ets_list_iter entityTypesGot_end =
    std::unique(s_entityTypesGot.begin(),
                s_entityTypesGot.end());

  // take the set difference
  std::set_difference(s_entityTypesWant.begin(),                    // [3]
                      entityTypesWant_end,                          // use new logical end
                      s_entityTypesGot.begin(),
                      entityTypesGot_end,
                      std::back_inserter(s_entityTypesDiff));    // [4]

  // [1] sorting: the 'set_difference' function template from
  // <algorithm> requires sorted ranges -- moreover here the
  // range is the entire container -- and note also that the
  // 'sort' function template from <algorithm> returns 'void'
  //
  // [2] all duplicates: to "remove" all (and not just
  // consecutive) duplicates, the container must first be sorted
  // -- note too that the discarded values are simply moved to
  // the end of the container and hence the need to use the "new
  // logical end" iterator (or invoke the 'erase' member function
  // for a permanent change)
  //
  // [3] set difference: yields, in set notation: Want / Got
  //
  // [4] results container: Josuttis (1999 p420) writes "the
  // caller must ensure that the destination range is big enough
  // or that insert iterators are used" -- the latter approach is
  // used here as also indicated in Josuttis (1999 p272)

  // return status information
  if ( s_entityTypesDiff.empty() )
    {
      return true;                              // all wants were got
    }
  else
    {
      s_logger->repx(logga::warn, "ungot wants total", s_entityTypesDiff.size());
      return false;                             // not all wants were got
    }
}

// STREAM INSERTION SUPPORT

// -------------------------------------------------------
//  MEMBER FUNCTION : streamOut
// -------------------------------------------------------
//  Description  : workhorse for overloaded operator<<
//  Role         : for streaming entities
//  Techniques   : ostreams
//  Status       : working but incomplete
// -------------------------------------------------------

std::ostream&
Entity::streamOut                           // support for overloaded operator<<
(std::ostream& os) const
{
  s_logger->repx(logga::warn, "stream insert helper not overridden", "");
  std::ios::fmtflags prior = os.flags();     // grab ostream state
  os << "    dummy output from an entity (not polymorphically overridden as expected)"
     << "\n";
  os.flags(prior);                          // reset ostream state
  return os;
}

// -------------------------------------------------------
//  CLASS           : FullEntity
// -------------------------------------------------------
//  Description  : stepping stone sub-entity
//  Role         : part of the full entity hierarchy
```

```
//  Techniques   : inheritance
//  Status       : complete
// ------------------------------------------------------------

// STATIC DEFINITIONS - with and without explicit initialization

int FullEntity::s_fullCount = 0;

// CREATORS

FullEntity::FullEntity
(const std::string entityId,
 Record&           record) :
  Entity(entityId, record),
  d_builtinRemark(record.tieSingle<std::string>("builtin-remark"))
{
  s_logger->repx(logga::adhc, "constructor call", getIdAndKind());
  isUniqueIdentifierFull();                  // char-by-char uniqueness
  ++s_fullCount;
  d_builtinRemark = "(not overwritten by entity author)";
}

FullEntity::~FullEntity()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
  --s_fullCount;
}

void
FullEntity::factoryInitialize()
{
  s_logger->repx(logga::xtra, "entity instance initialize", "full entity form");
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : listToVec <> (static)
// ------------------------------------------------------------
//  Description  : converts sub-entity lists to sub-entity vectors
//  Role         : explicit (rather than "lazy") multiple sub-entity linking
//  Techniques   : templates, downcasting, Boost.String_algo library
//  Requires     : friendship from 'Entity'
//  Status       : complete
//
//  Typical usage (for the hypothetical 'RedAnt' class):
//
//      const std::string&               d_red_ants;
//      std::vector<shared_ptr<RedAnt> > d_redAnts;
//
//      const int redAntsCnt
//        = FullEntity::listToVec<RedAnt>(d_red_ants, d_redAnts);
//
// ------------------------------------------------------------

template<typename E>                         // downcasting employed
const int                                    // returns number of elements in 'vec'
FullEntity::listToVec
(const std::string&           list,          // order is significant
 std::vector<shared_ptr<E> >& vec)           // pass by non-const reference, note the 'E'
{
  // initial logging, note that the hash-ifdef removes a
  // link-time error if the unit containing 'E' is not available
  // -- as may well be the case when unit testing
  const std::string msg = "entering member function, E type";
#ifdef _XUTEST
  s_logger->repx(logga::xtra, msg, "(not sought in unit tests)");
#else
  const std::string etype = xeona::demangle(typeid(E).name());
  s_logger->repx(logga::dbug, msg, etype);
#endif // _XUTEST

  // possibly report if 'list' is empty, noting that
  // 'xeona::rankNoData' is set by option '--nodata'
  if ( list.empty() )
    {
      s_logger->repx(logga::rankNoData, "list argument is empty", "");
    }

  // an empty vector is expected
  if ( ! vec.empty() )
    {
      s_logger->repx(logga::info, "vector argument is not empty", vec.size()); // [1]
```

```
      vec.clear();                          // remove all elements (but expecting empty)
    }
  // [1] CAUTION: this 'repx' used to be 'logga::warn' but the gateway
  // registeration process can make repeat calls to this function

  // CAUTION: an empty string (of size zero), when split,
  // produces a single empty element vector (of size one) -- this
  // is not helpful, hence the protection below keeps the split
  // list buffer empty in such cases

  // split the input, while noting the above caution
  std::vector<std::string> splitList;       // declare a split list buffer
  if ( ! list.empty() )
    {
      // split the list whilst compressing adjacent tokens
      boost::split(splitList, list, boost::is_any_of(" "), boost::token_compress_on);
    }

  // fill the vector
  BOOST_FOREACH( std::string s, splitList )
    {
      shared_ptr<Entity> sp;
      sp = Entity::retSharedPtr(s);         // requires friendship from 'Entity'
      if ( sp )                             // 'retSharedPtr' returns empty on failure
        {
          shared_ptr<E> sE;
          sE = dynamic_pointer_cast<E>(sp); // downcast to 'E'
          vec.push_back(sE);                // fill while maintaining order
        }
    }

  // integrity checks and reporting
  const int splitSize = splitList.size();   // input string duly split (unless empty)
  const int vecSize   = vec.size();         // resultant vector

  if ( splitSize > vecSize )                // hanging association/s encountered
    {
      std::ostringstream oss;
      oss << splitSize << " : " << vecSize;
      s_logger->repx(logga::rankNoData, "size mismatch, list : vector", oss.str());

      std::ostringstream put;
      std::ostringstream oss2;
      put << "    hanging association/s found" << "\n";
      BOOST_FOREACH( std::string s, splitList )
        {
          put  << "      " << s << "\n";
          oss2 << s << " ";                 // trailing space is later removed
        }
      s_logger->putx(logga::rankNoData, put);
      s_logger->addSmartBlank(logga::rankNoData);

      // throw if appropriate
      if ( xeona::nopro == false )          // meaning option '--krazy' not applied
        {
          s_logger->repx(logga::dbug, "will throw xeona::xem_data_issue", "");
          throw xeona::xem_data_issue("hanging association/s found",
                                      "offending identities",
                                      oss2.str());
        }
    }
  else if ( splitSize < vecSize )           // very strange result, should never be here
    {
      std::ostringstream oss;
      oss << splitSize << " : " << vecSize;
      s_logger->repx(logga::kill, "size mismatch, list : vector", oss.str());
    }

  // return vector size
  return vecSize;

} // function 'listToVec'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::putxId
// ----------------------------------------------------------

namespace xeona
{
  int
  putxId
```

```
  (const shared_ptr<Entity> e,
   const std::string&       comment)         // note empty string default in header
  {
    static logga::spLogger logger = logga::ptrLogStream();
    static int s_passes           = 0;
    static int s_fails            = 0;
    if ( e )
      {
        std::string buf = "";
        if ( ! comment.empty() ) buf = "  " + comment;
        const std::string id   = "  " + e->getIdAndKind();
#if 0 // 0 = add type information, 1 = skip
        const std::string type = "";                                        // cheap
#else
        const std::string type = "  " + xeona::demangle(typeid(*e).name());  // expensive
#endif // 0
        const std::string tag = "LOOP-";
        std::ostringstream put;
        put << boost::format("      %s%02d%s%s%s\n") % tag % ++s_passes % type % id % buf;
        logger->putx(logga::dbug, put);
      }
    else
      {
        logger->repx(logga::warn, "passed an empty/null entity, count", ++s_fails);
      }
    return s_fails;
  }

} // namespace 'xeona'

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------
//
//  Place all required template instantiations here.  That is,
//  place all required (or potentially required) header-declared
//  class and member function template instantiations at the end
//  of the implementation file.
//
//  (Alternatively, move the relevant definitions to the header
//  file.)
//
//  For simple information on template coding strategies, read
//  Cline (2006) section 35.  If this is insufficient for your
//  needs, refer to Lischner (2003 pp195-199) and Dattatri (2002
//  pp456-461).
//
//  Failure to properly instantiate a template will typically
//  result in a link-time error whenever a (constructor or
//  function) call is made.  The linker usually says: "undefined
//  reference to ...".  If no call is made, then the program
//  should compile normally and run without a murmur.
//
//    Cline, Marshall.  2006.  C++ FAQ lite.
//    [www.parashift.com/c++-faq-lite/templates.html]

//  CAUTION: regarding the 'listToVec' instantiations, the 'E'
//  needs to be fully defined, that is, the relevant header needs
//  to be hash-included -- otherwise the following error results
//  (noting that the header path has been resolved here):
//
//      /usr/include/c++/4.1.2/tr1/boost_shared_ptr.h:599:
//      error:
//      cannot dynamic_cast '__r->std::tr1::shared_ptr<Entity>::_M_ptr'
//      (of type 'class Entity* const') to type 'struct Xxxx*'
//      (target is not pointer or reference to complete type)

template
const int FullEntity::listToVec<AssetOperator>
(const std::string&, std::vector<shared_ptr<AssetOperator> >&);

template
const int FullEntity::listToVec<DemandJunction>
(const std::string&, std::vector<shared_ptr<DemandJunction> >&);

template
const int FullEntity::listToVec<DomainController>
(const std::string&, std::vector<shared_ptr<DomainController> >&);

template
const int FullEntity::listToVec<Gateway>
```

```
(const std::string&, std::vector<shared_ptr<Gateway> >&);

template
const int FullEntity::listToVec<LmpNode>
(const std::string&, std::vector<shared_ptr<LmpNode> >&);

template
const int FullEntity::listToVec<TechnicalAsset>
(const std::string&, std::vector<shared_ptr<TechnicalAsset> >&);

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : gateway.cc
//  file-create-date : Thu 23-Oct-2008 11:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : gateway entity which span commitment domains / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/gate.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "gate.h"             // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optprob.h"     // optimization sub-problem and key sub-classes
#include "../b/optgate.h"     // various OSPs for gateways
#include "../b/domcon.h"      // domain controller entity
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  FORWARD (PARTIAL) DECLARATIONS

namespace svif { class SolverIf; }           // member function argument

//  CODE

// ----------------------------------------------------------
//  CLASS           : Gateway
// ----------------------------------------------------------

//  STATIC DEFINITIONS

const double Gateway::s_inf = std::numeric_limits<double>::infinity();
const double Gateway::s_nil = std::numeric_limits<double>::quiet_NaN();

// CREATORS

Gateway::Gateway
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  Block(entityId, record),
  d_techCapacity(std::make_pair(s_nil , s_nil)),  // nonsensical value
```

```
  d_commCapacity(std::make_pair(s_nil , s_nil)),  // nonsensical value
  d_capacity(std::make_pair(s_nil, s_nil)),       // nonsensical value
  d_transaction(s_nil),                           // nonsensical value
  d_totalCost(s_nil),                             // nonsensical value
  d_dutyGol(-1),                                  // nonsensical value
  d_tilde(false),
  d_theta(false),
  d_star(false),
  d_hash(false)
{
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
}

Gateway::~Gateway()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ACCESSORS

bool Gateway::getTilde() const { return d_tilde; }
bool Gateway::getTheta() const { return d_theta; }
bool Gateway::getStar()  const { return d_star;  }
bool Gateway::getHash()  const { return d_hash;  }

bool
Gateway::getTransacted() const
{
  const bool status = xeona::isNan(d_transaction) ? false : true;
  s_logger->repx(logga::adhc, "transaction status", status);
  return status;
}

double
Gateway::getTransaction() const
{
  return d_transaction;
}

double
Gateway::getTotalCost() const
{
  s_logger->repx(logga::warn, "total cost probably not set", d_totalCost);
  return d_totalCost;
}

// MANIPULATORS

void
Gateway::unsetTransaction()
{
  s_logger->repx(logga::dbug, "entering member function", "");
  d_transaction = s_nil;
}

void
Gateway::setLowerCapacity
(const double lowerCta )
{
  d_capacity.first  = lowerCta;
}

void
Gateway::setUpperCapacity
(const double upperCta )
{
  d_capacity.second = upperCta;
}

bool
Gateway::markTilde()
{
  const bool prior = d_tilde;
  d_tilde          = true;
  return prior;
}

bool
Gateway::unmarkTilde()
{
  const bool prior = d_tilde;
```

```
  d_tilde          = false;
  return prior;
}

bool
Gateway::markTheta()
{
  const bool prior = d_theta;
  d_theta          = true;
  return prior;
}

bool
Gateway::unmarkTheta()
{
  const bool prior = d_theta;
  d_theta          = false;
  return prior;
}

bool
Gateway::markStar()
{
  const bool prior = d_star;
  d_star           = true;
  return prior;
}

bool
Gateway::unmarkStar()
{
  const bool prior = d_star;
  d_star           = false;
  return prior;
}

bool
Gateway::markHash()
{
  const bool prior = d_hash;
  d_hash           = true;
  return prior;
}

bool
Gateway::unmarkHash()
{
  const bool prior = d_hash;
  d_hash           = false;
  return prior;
}

void
Gateway::reset()
{
  s_logger->repx(logga::adhc, "entering member function", getIdAndKind());

  d_capacity    = std::make_pair(0.0, s_inf);     // sale (lower, upper) capacity
  d_transaction = s_nil;                          // sale quantity

  d_dutyGol     = -1;

  d_tilde       = false;                    // depth-first search label
  d_theta       = false;                    // capacity back-track lock
  d_star        = false;                    // transaction forward-track lock
  d_hash        = false;                    // fully-capacitated flag
}

// ----------------------------------------------------------
//  CLASS           : BuySide
// ----------------------------------------------------------

BuySide::BuySide
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  Gateway(entityId, record, commitmentModeSum),
  TicToc(commitmentModeSum),
  CostRegister(record),
  d_controller()
{
```

```
  s_logger->repx(logga::xtra, "constructor call", "");
}

BuySide::~BuySide()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

void
BuySide::registerDomain
(const std::string& actorId)
{
  s_logger->repx(logga::dbug, "entering buy-side member function", actorId);

  // CAUTION: friendship required for 'retSharedPtr'
  // note also the need to downcast
  d_controller = dynamic_pointer_cast<DomainController>(Entity::retSharedPtr(actorId));
}

void
BuySide::registerDomain
(shared_ptr<DomainController> domcon)
{
  s_logger->repx(logga::dbug,
                 "entering buy-side member function",
                 domcon->getIdAndKind());
  d_controller = domcon;
}

void
BuySide::initialize
(const int                   step,
 shared_ptr<svif::SolverIf> solver)          // solver instance passed thru
{
  s_logger->repx(logga::adhc, "entering member function", "");
  TicToc::initialize(step, solver);
  initializeBuySide(step, solver);
}

const int
BuySide::constrain                           // set constraints
(const xeona::DomainMode capacityMode)
{
  s_logger->repx(logga::adhc, "entering member function", "");
  const int buyDutyGol = constrainBuySide(capacityMode);
  d_dutyGol = buyDutyGol;
  return buyDutyGol;
}

void
BuySide::washup()
{
  s_logger->repx(logga::adhc, "entering member function", "");
  washupBuySide();
  loadRegisterBuySide(d_step);
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : recordTransaction
// ------------------------------------------------------------
//
//  Design notes
//
//      It might be better to make this function virtal and
//      service it from the various concrete classes -- instead
//      of capturing and then using 'd_dutyGol' directly with the
//      buy-side 'd_solver.'
//
// ------------------------------------------------------------

double
BuySide::recordTransaction()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // integrity checks
  if ( ! xeona::isNan(d_transaction) )
    {
      s_logger->repx(logga::warn, "transaction already set", d_transaction);
    }
```

```
  if ( d_dutyGol < 1 )
    {
      s_logger->repx(logga::warn, "nonsensical duty gol", d_dutyGol);
    }

  // obtain value from solver
  const double transaction = d_solver->getVarValue(d_dutyGol);

  s_logger->repx(logga::adhc, "transaction recorded", transaction);

  // update state
  d_transaction = transaction;

  // additional reporting as appropriate
  // YEEK 21 CODE (set by '--yeek')
  if ( xeona::yeek == 21 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put
        << "  gateway status" << "\n"
        << "    identifier  : " << getIdAndKind()                                  << "\n"
        << "    capacities  : " << d_capacity.first << " | " << d_capacity.second << "\n"
        << "    transaction : " << d_transaction                                   << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // return
  return d_transaction;
}

// ----------------------------------------------------------
//  CLASS           : SelSide
// ----------------------------------------------------------

SelSide::SelSide
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  Gateway(entityId, record, commitmentModeSum),
  TicToc(commitmentModeSum),
  CostRegister(record),
  d_controller()
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

SelSide::~SelSide()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

void
SelSide::registerDomain
(const std::string& actorId)
{
  s_logger->repx(logga::dbug, "entering sel-side member function", actorId);

  // CAUTION: friendship required for 'retSharedPtr', note also
  // the need to downcast (increase specialization)
  d_controller = dynamic_pointer_cast<DomainController>(Entity::retSharedPtr(actorId));
}

void
SelSide::registerDomain
(shared_ptr<DomainController> domcon)
{
  s_logger->repx(logga::dbug,
                 "entering buy-side member function",
                 domcon->getIdAndKind());
  d_controller = domcon;
}

void
SelSide::initialize
(const int                  step,
 shared_ptr<svif::SolverIf> solver)          // solver instance passed thru
{
  s_logger->repx(logga::adhc, "entering member function", "");
  TicToc::initialize(step, solver);
  initializeSelSide(step, solver);
```

```
}

const int
SelSide::constrain                          // set constraints
(const xeona::DomainMode capacityMode)
{
  s_logger->repx(logga::adhc, "entering member function", "");
  const int selDutyGol = constrainSelSide(capacityMode);
  d_dutyGol = selDutyGol;
  return selDutyGol;
}

void
SelSide::washup()
{
  s_logger->repx(logga::adhc, "entering member function", "");
  washupSelSide();
  loadRegisterSelSide(d_step);
}

// ----------------------------------------------------------
//  CLASS           : GateCom <>
// ----------------------------------------------------------

template <typename C>
GateCom<C>::GateCom
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  Gateway(entityId, record, commitmentModeSum),      // CAUTION: virtual base class init
  BuySide(entityId, record, commitmentModeSum),
  SelSide(entityId, record, commitmentModeSum),
  // tied quantities
  d_cable(Cable<C>::create
          (entityId,                          // me
           record.tieSingle<std::string>("socket"),
           record.tieSingle<std::string>("common-commodity"))),
  d_socket(Socket<C>::create
           (entityId,                         // me
            "sock-1",                         // hardcoded socket label
            record.tieSingle<std::string>("common-commodity")))
  // internal quantities
{
  s_logger->repx(logga::dbug, "constructor call", "");
}

template <typename C>
GateCom<C>::~GateCom()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

template <typename C>
void
GateCom<C>::unitTestSay()
{
#ifdef _XUTEST
  // no code is correct
#endif // _XUTEST
}

template <typename C>
shared_ptr<DomainController>
GateCom<C>::hop
(const shared_ptr<DomainController> me)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // additional reporting as appropriate
  // YEEK 19 CODE (set by '--yeek')
  if ( xeona::yeek == 19 || xeona::yeek == 26 || xeona::yeek == 1 )
    {
      std::string direction = "(null domain controller)";
      if ( me )
        {
          if ( SelSide::d_controller == me ) direction = "buyward";
          else                               direction = "selward";
        }
      std::ostringstream put;
      put << "  function " << __func__                                    << "\n"
```

```
          << "    host identifier : " << getIdAndKind()                        << "\n"
          << "    me (argument)   : " << me->getIdAndKind()                    << "\n"
          << "    buy-side domain : " << BuySide::d_controller->getIdAndKind()  << "\n"
          << "    sel-side domain : " << SelSide::d_controller->getIdAndKind()  << "\n"
          << "    direction       : " << direction                             << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // active code
  if ( me )                                   // not empty or not holding null pointer
    {
      // simply return my counterparty
      if ( SelSide::d_controller == me )
        return BuySide::d_controller;
      else
        return SelSide::d_controller;
    }
  else                                        // empty or holding null pointer
    {
      // log warning and return an empty shared pointer
      s_logger->repx(logga::warn, "empty or null shared pointer given", "");
      return shared_ptr<DomainController>();
    }
}

template <typename C>
shared_ptr<Block>
GateCom<C>::getDemander() const
{
  shared_ptr<Entity> entity = d_socket->getPartner();
  shared_ptr<Block>  block  = dynamic_pointer_cast<Block>(entity);
  if ( block == 0 ) s_logger->repx(logga::warn, "Block cast failed", "");
  return block;
}

template <typename C>
shared_ptr<Block>
GateCom<C>::getSupplier() const
{
  shared_ptr<Entity> entity = d_cable->getPartner();
  shared_ptr<Block>  block  = dynamic_pointer_cast<Block>(entity);
  if ( block == 0 ) s_logger->repx(logga::warn, "Block cast failed", "");
  return block;
}

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// Commodity typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class GateCom <>

template GateCom<Oxid>::GateCom(const std::string, Record&, const int);
template GateCom<Cert>::GateCom(const std::string, Record&, const int);
template GateCom<Cseq>::GateCom(const std::string, Record&, const int);
template GateCom<Elec>::GateCom(const std::string, Record&, const int);
template GateCom<Work>::GateCom(const std::string, Record&, const int);
template GateCom<Heat>::GateCom(const std::string, Record&, const int);
template GateCom<Thrm>::GateCom(const std::string, Record&, const int);
template GateCom<Fund>::GateCom(const std::string, Record&, const int);

template GateCom<Oxid>::~GateCom();
template GateCom<Cert>::~GateCom();
template GateCom<Cseq>::~GateCom();
```

```
template GateCom<Elec>::~GateCom();
template GateCom<Work>::~GateCom();
template GateCom<Heat>::~GateCom();
template GateCom<Thrm>::~GateCom();
template GateCom<Fund>::~GateCom();

template void GateCom<Oxid>::unitTestSay();
template void GateCom<Cert>::unitTestSay();
template void GateCom<Cseq>::unitTestSay();
template void GateCom<Elec>::unitTestSay();
template void GateCom<Work>::unitTestSay();
template void GateCom<Heat>::unitTestSay();
template void GateCom<Thrm>::unitTestSay();
template void GateCom<Fund>::unitTestSay();

template shared_ptr<Block> GateCom<Oxid>::getDemander() const;
template shared_ptr<Block> GateCom<Cert>::getDemander() const;
template shared_ptr<Block> GateCom<Cseq>::getDemander() const;
template shared_ptr<Block> GateCom<Elec>::getDemander() const;
template shared_ptr<Block> GateCom<Work>::getDemander() const;
template shared_ptr<Block> GateCom<Heat>::getDemander() const;
template shared_ptr<Block> GateCom<Thrm>::getDemander() const;
template shared_ptr<Block> GateCom<Fund>::getDemander() const;

template shared_ptr<Block> GateCom<Oxid>::getSupplier() const;
template shared_ptr<Block> GateCom<Cert>::getSupplier() const;
template shared_ptr<Block> GateCom<Cseq>::getSupplier() const;
template shared_ptr<Block> GateCom<Elec>::getSupplier() const;
template shared_ptr<Block> GateCom<Work>::getSupplier() const;
template shared_ptr<Block> GateCom<Heat>::getSupplier() const;
template shared_ptr<Block> GateCom<Thrm>::getSupplier() const;
template shared_ptr<Block> GateCom<Fund>::getSupplier() const;

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : gate1.cc
//  file-create-date : Wed 15-Apr-2009 21:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete gateways 1 - stated tariff / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/gate01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "gate01.h"           // companion header for this file (place first)

#include "../c/util3.h"       // free functions for floating point comparison
#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optprob.h"     // optimization sub-problem and key sub-classes
#include "../b/optgate.h"     // various OSPs for gateways
#include "../b/domcon.h"      // domain controller entity
#include "../b/commods.h"     // commodities hierarchy
#include "../b/bandtaf.h"     // banded tariff set and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <utility>            // STL pair, make_pair()

//  CODE

// ----------------------------------------------------------
//  CLASS           : GateStatedTariff <>
// ----------------------------------------------------------
//
//  CAUTION: name resolution within member functions
//
//      A 'using' declaration (as opposed to a 'using' directive)
//      can only be used at class scope and not at function scope
//      -- hence the need to use fully qualify certain
//      identifiers within member functions.  Most notably:
//
//          d_step
//          d_commitmentMode
//          d_solver
//
// ----------------------------------------------------------

template <typename C>
```

```
GateStatedTariff<C>::GateStatedTariff
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  Gateway(entityId, record, commitmentModeSum),   // CAUTION: virtual base class init
  GateCom<C>(entityId, record, commitmentModeSum),
  // tied quantities
  d_tariffsets(record.tieTimeseries<std::string>("tariffsets")),
  d_definedCapacitys(record.tieTimeseries<double>("defined-capacitys")),
  d_quantitys(record.tieTimeseries<double>("quantitys")),
  d_marginalPrices(record.tieTimeseries<double>("marginal-prices")),
  d_totalCosts(record.tieTimeseries<double>("total-costs")),
  // internal quantities
  d_tariffset(),                                  // empty shared pointer
  d_capSel(),
  d_capBuy(),
  d_ots(),
  d_ofr()
{
  s_logger->repx(logga::adhc, "constructor call", getIdAndKind());
  GateCom<C>::d_builtinRemark = "beta";           // CAUTION: note scope resolution
}

template <typename C>
GateStatedTariff<C>::~GateStatedTariff()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : establish
// ------------------------------------------------------------

// the 'd_tariffset' variable is filled for each interval under
// 'initializeBuySide' -- so it does not need to be generated for
// the whole horizon here

template <typename C>
void
GateStatedTariff<C>::establish()
{
  // additional reporting as appropriate
  // YEEK 18 CODE (set by '--yeek')
  if ( xeona::yeek == 18 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  gateway establish call"           << "\n"
          << "    type     : GateStatedTariff<C>" << "\n"
          << "    identity : " << getIdAndKind()  << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }
  else
    {
      s_logger->repx(logga::adhc, "entering member function", "nothing to do");
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : conclude
// ------------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::conclude()
{
      s_logger->repx(logga::adhc, "entering member function", "nothing to do");
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : initializeBuySide
// ------------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::initializeBuySide
(const int                  step,
 shared_ptr<svif::SolverIf> solver)
{
  s_logger->repx(logga::adhc, "entering member function", "");
```

```
  // reset and then load the tariffset
  d_tariffset.reset(new BandedTariffSet());
  d_tariffset->pushString(d_tariffsets->at(step));
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : initializeSelSide
// -----------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::initializeSelSide
(const int                  step,
 shared_ptr<svif::SolverIf> solver)
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // do nothing is correct
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : constrainBuySide
// -----------------------------------------------------------

// resets and fills 'd_tariffset' of class 'BandedTariffSet'
//
// resets and fills 'd_capBuy'    of class 'QanTechCapacity'
// resets and fills 'd_ofr'       of class 'OfrTariffSet'
//
// the tariff set is only used here and is not used by 'constrainSelSide'

template <typename C>
const int
GateStatedTariff<C>::constrainBuySide
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "GateStatedTariff<C>");

  // define some local variables for convenience
  const int step                     = GateCom<C>::BuySide::d_step;
  shared_ptr<svif::SolverIf> solver = GateCom<C>::BuySide::d_solver;
  xeona::DomainMode commitmentMode  = GateCom<C>::BuySide::d_commitmentMode;

  // TECHNICAL CAPACITY OSP

  // create and fill a label object
  const std::string gateId = getIdentifier();
  Label capLabel(gateId);
  capLabel << "capset-capacity";

  // locate key values
  const double lower = 0.0;
  const double upper = d_definedCapacitys->at(step);
  d_capacity = std::make_pair(lower, upper);      // strictly 'Gateway::d_capacity'

  // recreate new operations OSP of the required type
  d_capBuy.reset(new QanTechCapacity(solver, commitmentMode, ""));
  d_capBuy->loadOspLabel(capLabel.str());

  // define global col/cols for internal use
  int buyGol = 0;

  // upload the engineering (a simple upper bound onto the buy-side)
  boost::tie(buyGol) = d_capBuy->uploadCapacity(d_capacity);

  // not required: upload specific costs
  // not required: store some values ('d_capacity' set by 'constrainSelSide')

  // TARIFFSET OSP

  // create and fill a label object
  Label ofrLabel(gateId);
  ofrLabel << "offer";

  // recreate new operations OSP of the required type
  d_ofr.reset(new OfrTariffSet(solver, commitmentMode, ""));
  d_ofr->loadOspLabel(ofrLabel.str());

  // define global col/cols for internal use
  int ofrGol = 0;
```

```
  // upload the tariffset
  Gateway::bounded_type capacity = d_capacity;  // CAUTION: cannot use "d_capacity.second"
  boost::tie(ofrGol) = d_ofr->uploadTariffSet(d_tariffset, capacity.second);

  // not required: upload specific costs
  // not required: bind global cols to the relevant interfaces
  // not required: store some values

  // BINDINGS

  // OSP couple call (from unit 'b/optprob')
  xeona::couple(solver,
                buyGol,
                ofrGol,
                "buy-side.couple");

  // bind global cols to the relevant interfaces
  d_socket->bindOsp(solver, buyGol);

  // RETURN

  // return the duty gol
  s_logger->repx(logga::dbug, "leaving member function, returning", buyGol);
  return buyGol;

} // function 'GateStatedTariff<C>::constrainBuySide'

// ------------------------------------------------------------
//  MEMBER FUNCTION : constrainSelSide
// ------------------------------------------------------------

// resets and fills 'd_capSel' of class 'QanTechCapacity'
// resets and fills 'd_ots'    of class 'QanObligToSupply'

template <typename C>
const int
GateStatedTariff<C>::constrainSelSide
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "GateStatedTariff<C>");

  // define some local variables for convenience
  const int step                    = GateCom<C>::SelSide::d_step;
  shared_ptr<svif::SolverIf> solver = GateCom<C>::SelSide::d_solver;
  xeona::DomainMode commitmentMode  = GateCom<C>::SelSide::d_commitmentMode;

  // TECHNICAL CAPACITY OSP

  // create and fill a label object
  const std::string gateId = getIdentifier();
  Label opsLabel(gateId);
  opsLabel << "tech-capacity";

  // locate key values
  const double lower = 0.0;
  const double upper = d_definedCapacitys->at(step);
  d_capacity = std::make_pair(lower, upper); // strictly 'Gateway::d_capacity'

  // recreate new operations OSP of the required type
  d_capSel.reset(new QanTechCapacity(solver, commitmentMode, ""));
  d_capSel->loadOspLabel(opsLabel.str());

  // define global col/cols for internal use
  int selGol = 0;

  // upload the engineering (a simple upper bound onto the sel-side)
  boost::tie(selGol) = d_capSel->uploadCapacity(d_capacity);

  // not required: upload specific costs

  // store capacity value as a technical capacity
  d_techCapacity = d_capacity;              // strictly 'Gateway::d_techCapacity'

  // OBLIGATION OSP

  // create and fill a label object
  Label otsLabel(gateId);
  otsLabel << "tech-capacity";
```

```
  // recreate new operations OSP of the required type
  d_ots.reset(new QanObligToSupply(solver, commitmentMode, ""));
  d_ots->loadOspLabel(otsLabel.str());

  // define global col/cols for internal use
  int otsGol = 0;

  // upload the supply obligation
  boost::tie(otsGol) = d_ots->uploadObligation(d_transaction);

  // not required: upload specific costs
  // not required: bind global cols to the relevant interfaces
  // not required: store some values

  // BINDINGS

  // OSP couple call (from unit 'b/optprob')
  xeona::couple(solver,
                selGol,
                otsGol,
                "buy-side.couple");

  // bind global cols to the relevant interfaces
  d_cable->bindOsp(solver, selGol);

  // RETURN

  // return the duty gol
  s_logger->repx(logga::dbug, "leaving member function, returning", selGol);
  return selGol;

} // function 'GateStatedTariff<C>::constrainSelSide'

// ------------------------------------------------------------
//  MEMBER FUNCTION : washupBuySide
// ------------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::washupBuySide()
{
  s_logger->repx(logga::adhc, "entering member function", getIdAndKind());
  s_logger->repx(logga::adhc, "any tasks currently left to sel-side", "");

} // function 'GateStatedTariff<C>::washupBuySide'

// ------------------------------------------------------------
//  MEMBER FUNCTION : washupSelSide
// ------------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::washupSelSide()
{
  // define some local variables for convenience
  const int step = GateCom<C>::SelSide::d_step;

  // interval (time span in seconds)
  const int interval = Entity::getHorizonInterval();

  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", step);

  // results recovery
  double marginalPrice  = -1.0;              // unlikely value
  double fixedComponent = -1.0;              // unlikely value
  double totalPrice     = -1.0;              // unlikely value
  double transaction    = -1.0;              // nonsensical value
  boost::tie(marginalPrice,
             fixedComponent,
             totalPrice,
             transaction) = d_ofr->downloadSolution(interval);
  double duty = -1.0;                        // nonsensical value
  boost::tie(duty) = d_capSel->downloadSolution();

  // grab capacities
  Gateway::bounded_type capacity = d_capacity;  // CAUTION: cannot use "d_capacity.second"
  double lowerCapacity = capacity.first;
  double upperCapacity = capacity.second;

  // additional reporting as appropriate
```

```
  // YEEK 32 CODE (set by '--yeek')
  if ( xeona::yeek == 32 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  gateway information"                                    << "\n"
          << "    recovered"                                            << "\n"
          << "        marginal price             : " << marginalPrice    << "\n"
          << "        fixed component            : " << fixedComponent   << "\n"
          << "        total price                : " << totalPrice       << "\n"
          << "        transaction (OfrTariffSet) : " << transaction      << "\n"
          << "        duty (QanTechCapacity)     : " << duty             << "\n"
          << "    recorded"                                             << "\n"
          << "        transaction (d_transaction) : " << d_transaction   << "\n"
          << "        lower capacity (d_capacity) : " << lowerCapacity   << "\n"
          << "        upper capacity (d_capacity) : " << upperCapacity   << "\n"
          << ""                                                         << "\n"
          << boost::format("    transaction : %22.12f") %   transaction   << "\n"
          << boost::format("  d_transaction : %22.12f") % d_transaction   << "\n";

      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // integrity check, note that strict equality is NOT okay
  if ( transaction == d_transaction )
    {
      // do nothing is correct -- in addition this conditional
      // obviates the need for further tests
    }
  else if ( ! xeona::almostEqual(transaction, d_transaction, xeona::numic) )  // "numeric"
    {
      std::ostringstream oss;
      oss << transaction << " : " << d_transaction;
      s_logger->repx(logga::warn, "transaction mismatch (try yeek 32)", oss.str());
      s_logger->repx(logga::dbug, "almostEqual fail level xeona::numic", "");
    }
  else if ( ! xeona::almostEqual(transaction, d_transaction, xeona::tight) )
    {
      std::ostringstream oss;
      oss << transaction << " : " << d_transaction;
      s_logger->repx(logga::warn, "transaction mismatch (try yeek 32)", oss.str());
      s_logger->repx(logga::dbug, "almostEqual fail level xeona::tight", "");
      if ( xeona::nopro == false )           // meaning option '--krazy' not applied
        {
          s_logger->repx(logga::dbug, "use --krazy to omit kill", "");
          s_logger->repx(logga::kill, "significant transaction mismatch", "");
        }
      else
        {
          s_logger->repx(logga::warn, "defensive code being omitted", "");

        }
    }

  // store entity state information
  d_quantitys->at(step)      = transaction;
  d_marginalPrices->at(step) = marginalPrice;
  d_totalCosts->at(step)     = totalPrice;

  // embedded costs processing, outer call also updates instance data
  // (not required as no costs are deemed to arise at a gateway)

  // store some on-the-fly statistics
  d_sizeStats(capacity.second);              // functor provided by class 'Block'
  d_dutyStats(d_transaction);                // functor provided by class 'Block'

} // member function 'GateStatedTariff<C>::washupSelSide'

// ----------------------------------------------------------
//  MEMBER FUNCTION : loadRegisterBuySide
// ----------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::loadRegisterBuySide
(const int step)
{
  s_logger->repx(logga::adhc, "entering member function", getIdAndKind());

  // short-run costs processing, outer call also updates instance data
  GateCom<C>::BuySide::updateShortrunCosts   // polymorphic 'CostRegister' call
```

```
    (step,                                  // provided by class 'TicToc'
     d_ofr->downloadShortrunCosts());

} // function 'GateStatedTariff<C>::loadRegisterBuySide'

// ----------------------------------------------------------
//  MEMBER FUNCTION : loadRegisterSelSide
// ----------------------------------------------------------

template <typename C>
void
GateStatedTariff<C>::loadRegisterSelSide
(const int step)
{
  s_logger->repx(logga::adhc, "entering member function", getIdAndKind());

  // CAUTION: the short-run costs need to be made negative
  // because these costs need to be treated as revenues

  // short-run costs processing, outer call also updates instance data
  GateCom<C>::BuySide::updateShortrunCosts   // polymorphic 'CostRegister' call
    (step,                                   // provided by class 'TicToc'
     -d_ofr->downloadShortrunCosts());       // CAUTION: unary minus

} // function 'GateStatedTariff<C>::loadRegisterSelSide'

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// Commodity typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class GateStatedTariff <>

template GateStatedTariff<Oxid>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Cert>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Cseq>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Elec>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Work>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Heat>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Thrm>::GateStatedTariff(const std::string, Record&, const int);
template GateStatedTariff<Fund>::GateStatedTariff(const std::string, Record&, const int);

template GateStatedTariff<Oxid>::~GateStatedTariff();
template GateStatedTariff<Cert>::~GateStatedTariff();
template GateStatedTariff<Cseq>::~GateStatedTariff();
template GateStatedTariff<Elec>::~GateStatedTariff();
template GateStatedTariff<Work>::~GateStatedTariff();
template GateStatedTariff<Heat>::~GateStatedTariff();
template GateStatedTariff<Thrm>::~GateStatedTariff();
template GateStatedTariff<Fund>::~GateStatedTariff();

template void      GateStatedTariff<Oxid>::establish();
template void      GateStatedTariff<Cert>::establish();
template void      GateStatedTariff<Cseq>::establish();
template void      GateStatedTariff<Elec>::establish();
template void      GateStatedTariff<Work>::establish();
template void      GateStatedTariff<Heat>::establish();
template void      GateStatedTariff<Thrm>::establish();
template void      GateStatedTariff<Fund>::establish();

template const int GateStatedTariff<Oxid>::constrainBuySide(const xeona::DomainMode);
template const int GateStatedTariff<Cert>::constrainBuySide(const xeona::DomainMode);
template const int GateStatedTariff<Cseq>::constrainBuySide(const xeona::DomainMode);
template const int GateStatedTariff<Elec>::constrainBuySide(const xeona::DomainMode);
template const int GateStatedTariff<Work>::constrainBuySide(const xeona::DomainMode);
```

```
template const int GateStatedTariff<Heat>::constrainBuySide(const xeona::DomainMode);
template const int GateStatedTariff<Thrm>::constrainBuySide(const xeona::DomainMode);
template const int GateStatedTariff<Fund>::constrainBuySide(const xeona::DomainMode);

template const int GateStatedTariff<Oxid>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Cert>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Cseq>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Elec>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Work>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Heat>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Thrm>::constrainSelSide(const xeona::DomainMode);
template const int GateStatedTariff<Fund>::constrainSelSide(const xeona::DomainMode);

template void      GateStatedTariff<Oxid>::washupBuySide();
template void      GateStatedTariff<Cert>::washupBuySide();
template void      GateStatedTariff<Cseq>::washupBuySide();
template void      GateStatedTariff<Elec>::washupBuySide();
template void      GateStatedTariff<Work>::washupBuySide();
template void      GateStatedTariff<Heat>::washupBuySide();
template void      GateStatedTariff<Thrm>::washupBuySide();
template void      GateStatedTariff<Fund>::washupBuySide();

template void      GateStatedTariff<Oxid>::washupSelSide();
template void      GateStatedTariff<Cert>::washupSelSide();
template void      GateStatedTariff<Cseq>::washupSelSide();
template void      GateStatedTariff<Elec>::washupSelSide();
template void      GateStatedTariff<Work>::washupSelSide();
template void      GateStatedTariff<Heat>::washupSelSide();
template void      GateStatedTariff<Thrm>::washupSelSide();
template void      GateStatedTariff<Fund>::washupSelSide();

// define a preprocessor macro for convenience when making
// explicit template instantiations
//
// CAUTION: Lischner (2003 p276) says "A backslash (\) at the end
// of the line continues the directive onto the subsequent line."

#define ARGS_1 \
shared_ptr<svif::SolverIf>, \
const xeona::DomainMode, \
const xeona::DomainMode, \
const std::string

// EXPLANATION: at one point, long template instantiations were
// shortened using this macro -- this facility, however, is no
// longer needed but the macro code remains for the time being

// macro no longer needed!
#undef ARGS_1 // see this file

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : junc.cc
//  file-create-date : Mon 26-Oct-2009 12:44 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : demand split/join junction entity // implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/junc.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "junc.h"             // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  MEMBER FUNCTION : DemandJunction
// ----------------------------------------------------------

DemandJunction::DemandJunction
(const std::string entityId,
 Record&           record) :
  Block(entityId, record),
  TicToc(xeona::e_commitmentModes)           // CAUTION: value set here, take note
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~DemandJunction
// ----------------------------------------------------------

DemandJunction::~DemandJunction()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : junc01.cc
//  file-create-date : Mon 26-Oct-2009 12:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete demand split junctions / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/junc01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "junc01.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optjunc.h"     // operations OSPs for junctions
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : JuncDemand2Split <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : JuncDemand2Split
// ----------------------------------------------------------

template <typename C>
JuncDemand2Split<C>::JuncDemand2Split
(const std::string entityId,
 Record&           record) :
  DemandJunction(entityId, record),
  d_junctionCommodity(record.tieSingle<std::string>("junction-commodity")),
  d_cable1(Cable<C>::create
           (entityId,                          // me
            record.tieSingle<std::string>("socket-1"),
            d_junctionCommodity)),             // common value
  d_cable2(Cable<C>::create
           (entityId,                          // me
            record.tieSingle<std::string>("socket-2"),
            d_junctionCommodity)),             // common value
  d_socket(Socket<C>::create
```

```
           (entityId,                         // me
            "sock-1",                         // hardcoded socket label
            d_junctionCommodity)),            // common value
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~JuncDemand2Split
// ----------------------------------------------------------

template <typename C>
JuncDemand2Split<C>::~JuncDemand2Split()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------

template <typename C>
const int                                     // zero
JuncDemand2Split<C>::constrain
(const xeona::DomainMode capacityMode)        // not used
{
#if 0 // 0 = normal reporting, 1 = extended reporting
  // initial extended reporting
  const std::string subtype = xeona::demangle(typeid(*this).name());  // CAUTION: deref
  s_logger->repx(logga::adhc, "entering member function", subtype);
#else
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "JuncDemand2Split");
#endif // 0

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new JncSplit2(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int outputGol  = -1;                    // nonsensical value
  int factor1Gol = -1;                    // nonsensical value
  int factor2Gol = -1;                    // nonsensical value

  // upload the engineering
  boost::tie(outputGol,
             factor1Gol,
             factor2Gol)
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable1->bindOsp(d_solver, factor1Gol);
  d_cable2->bindOsp(d_solver, factor2Gol);
  d_socket->bindOsp(d_solver, outputGol);

  // return zero
  return 0;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

template <typename C>
void
JuncDemand2Split<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
```

```
  double output      = -1.0;                // nonsensical value
  boost::tie(output) = d_ops->downloadSolution();

  // store some on-the-fly statistics
  d_dutyStats(output);                      // functor provided by class 'Block'
  d_sizeStats();                            // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// class 'Commodity' typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class 'JuncDemand2Split<>'

template JuncDemand2Split<Oxid>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Cert>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Cseq>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Elec>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Work>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Heat>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Thrm>::JuncDemand2Split(const std::string, Record&);
template JuncDemand2Split<Fund>::JuncDemand2Split(const std::string, Record&);

template JuncDemand2Split<Oxid>::~JuncDemand2Split();
template JuncDemand2Split<Cert>::~JuncDemand2Split();
template JuncDemand2Split<Cseq>::~JuncDemand2Split();
template JuncDemand2Split<Elec>::~JuncDemand2Split();
template JuncDemand2Split<Work>::~JuncDemand2Split();
template JuncDemand2Split<Heat>::~JuncDemand2Split();
template JuncDemand2Split<Thrm>::~JuncDemand2Split();
template JuncDemand2Split<Fund>::~JuncDemand2Split();

template const int JuncDemand2Split<Oxid>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Cert>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Cseq>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Elec>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Work>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Heat>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Thrm>::constrain(const xeona::DomainMode);
template const int JuncDemand2Split<Fund>::constrain(const xeona::DomainMode);

template void JuncDemand2Split<Oxid>::washup();
template void JuncDemand2Split<Cert>::washup();
template void JuncDemand2Split<Cseq>::washup();
template void JuncDemand2Split<Elec>::washup();
template void JuncDemand2Split<Work>::washup();
template void JuncDemand2Split<Heat>::washup();
template void JuncDemand2Split<Thrm>::washup();
template void JuncDemand2Split<Fund>::washup();

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : junc02.cc
//  file-create-date : Tue 27-Oct-2009 09:18 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete demand join junctions / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/junc02.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "junc02.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optjunc.h"     // operations OSPs for junctions
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  MEMBER FUNCTION : JuncDemand2Join
// ----------------------------------------------------------

template <typename C>
JuncDemand2Join<C>::JuncDemand2Join
(const std::string entityId,
 Record&           record) :
  DemandJunction(entityId, record),
  d_junctionCommodity(record.tieSingle<std::string>("junction-commodity")),
  d_cable(Cable<C>::create
          (entityId,                          // me
           record.tieSingle<std::string>("socket-1"),
           d_junctionCommodity)),           // common value
  d_socket1(Socket<C>::create
            (entityId,                        // me
             "sock-1",                        // hardcoded socket label
             d_junctionCommodity)),           // common value
  d_socket2(Socket<C>::create
            (entityId,                        // me
             "sock-2",                        // hardcoded socket label
             d_junctionCommodity)),           // common value
  d_ops()
```

```
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~JuncDemand2Join
// ----------------------------------------------------------

template <typename C>
JuncDemand2Join<C>::~JuncDemand2Join()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------

template <typename C>
const int                                   // zero
JuncDemand2Join<C>::constrain
(const xeona::DomainMode capacityMode)      // not used
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "JuncDemand2Join");

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new JncJoin2(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int factorGol   = -1;                     // nonsensical value
  int output1Gol  = -1;                     // nonsensical value
  int output2Gol  = -1;                     // nonsensical value

  // upload the engineering
  boost::tie(factorGol,
             output1Gol,
             output2Gol)
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable  ->bindOsp(d_solver, factorGol);
  d_socket1->bindOsp(d_solver, output1Gol);
  d_socket2->bindOsp(d_solver, output2Gol);

  // return zero
  return 0;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

template <typename C>
void
JuncDemand2Join<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double output      = -1.0;                // nonsensical value
  boost::tie(output) = d_ops->downloadSolution();

  // store some on-the-fly statistics
  d_dutyStats(output);                      // functor provided by class 'Block'
  d_sizeStats();                            // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
```

```
// ----------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// class 'Commodity' typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class 'JuncDemand2Join<>'

template JuncDemand2Join<Oxid>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Cert>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Cseq>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Elec>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Work>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Heat>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Thrm>::JuncDemand2Join(const std::string, Record&);
template JuncDemand2Join<Fund>::JuncDemand2Join(const std::string, Record&);

template JuncDemand2Join<Oxid>::~JuncDemand2Join();
template JuncDemand2Join<Cert>::~JuncDemand2Join();
template JuncDemand2Join<Cseq>::~JuncDemand2Join();
template JuncDemand2Join<Elec>::~JuncDemand2Join();
template JuncDemand2Join<Work>::~JuncDemand2Join();
template JuncDemand2Join<Heat>::~JuncDemand2Join();
template JuncDemand2Join<Thrm>::~JuncDemand2Join();
template JuncDemand2Join<Fund>::~JuncDemand2Join();

template const int JuncDemand2Join<Oxid>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Cert>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Cseq>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Elec>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Work>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Heat>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Thrm>::constrain(const xeona::DomainMode);
template const int JuncDemand2Join<Fund>::constrain(const xeona::DomainMode);

template void JuncDemand2Join<Oxid>::washup();
template void JuncDemand2Join<Cert>::washup();
template void JuncDemand2Join<Cseq>::washup();
template void JuncDemand2Join<Elec>::washup();
template void JuncDemand2Join<Work>::washup();
template void JuncDemand2Join<Heat>::washup();
template void JuncDemand2Join<Thrm>::washup();
template void JuncDemand2Join<Fund>::washup();

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : lmpbid.cc
//  file-create-date : Fri 17-Oct-2008 12:53 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : LMP auction bidset and support / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/lmpbid.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "lmpbid.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <utility>            // STL pair, make_pair()
#include <vector>             // STL sequence container

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

//  CODE

// ------------------------------------------------------------
//  FUNCTION        : ::bid_less
// ------------------------------------------------------------
//  Description  : custom STL comparison predicate
//  Role         : sort insertion of individual bids within a given bidset
//  Techniques   : STL algorithms, unnamed namespace (file local scope)
//  Status       : working
// ------------------------------------------------------------

namespace
{
  bool
  bid_less
  (std::pair<double, double> one,
   std::pair<double, double> two)
  {
    return one.second < two.second;          // compares second element
  }
} // unnamed namespace

// ------------------------------------------------------------
```

```
//  CLASS           : LmpBidSet
// ---------------------------------------------------------
//  Description  : bidset abstraction for LMP (nodal) auctions
//  Role         : used to prepare, hold, and unpack a bidset
//  Techniques   : stand-alone class, 'std::pair', 'std::upper_bound'
//  Status       : complete but not tested
//
//  Design notes
//
//      Overview
//
//          A bidset is a sorted collection of (band, price)
//          'std::pair's held in a 'std::vector.'
//
//          The sort order is on ascending price.  If two bid
//          pairs have equal price, then their order of
//          appearance is preserved.
//
//      Input format
//
//          In terms of input, the 'xeona' model more generally
//          expects:
//
//            * a commodity band in [W]
//            * a unit price in [CUR/J]
//
//          The 'load' call expects a single string with the
//          following formatting rules:
//
//            * field order as given above
//            * bid separator as defined by 'xeona::modelBidDelim'
//            * band/price separator is space character
//
//          The individual bids need not be sorted in any way.
//          Typical input might be (the first bid equates to 4OMW
//          prices at about 10c/kWh):
//
//              "40.00e+06 28.00e-09 * 20.00e+06 40.00e-09"
//
// ---------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger LmpBidSet::s_logger = logga::ptrLogStream();

// CREATORS

// ---------------------------------------------------------
//  MEMBER FUNCTION : LmpBidSet
// ---------------------------------------------------------

LmpBidSet::LmpBidSet
(std::string label) :
  d_label(label),
  d_bidset(),
  d_negate(false)
{
  s_logger->repx(logga::xtra, "constructor call, label", d_label);
}

// ---------------------------------------------------------
//  MEMBER FUNCTION : ~LmpBidSet
// ---------------------------------------------------------

LmpBidSet::~LmpBidSet()
{
  s_logger->repx(logga::adhc, "destructor call", d_label);
}

// ---------------------------------------------------------
//  MEMBER FUNCTION : copy constructor
// ---------------------------------------------------------

// explicit definition required because of friendship declaration

LmpBidSet::LmpBidSet
(const LmpBidSet& orig) :
  d_label(orig.d_label),
  d_bidset(orig.d_bidset),
  d_negate(orig.d_negate)
{
```

```
  s_logger->repx(logga::adhc, "copy constructor call", d_label);
}

// UNARY OPERATORS

// -----------------------------------------------------------
//  MEMBER FUNCTION : operator+= (price increment)
// -----------------------------------------------------------

LmpBidSet&
LmpBidSet::operator+=
(const double& other)
{
  BOOST_FOREACH( bid_type& bid, d_bidset )   // CAUTION: typedef and reference essential
    {
      bid.second += other;
    }
  return *this;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : operator-= (price decrement)
// -----------------------------------------------------------

LmpBidSet&
LmpBidSet::operator-=
(const double& other)
{
  BOOST_FOREACH( bid_type& bid, d_bidset )   // CAUTION: typedef and reference essential
    {
      bid.second -= other;
    }
  return *this;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : operator*= (price multiplier)
// -----------------------------------------------------------

LmpBidSet&
LmpBidSet::operator*=
(const double& other)
{
  BOOST_FOREACH( bid_type& bid, d_bidset )   // CAUTION: typedef and reference essential
    {
      bid.second *= other;
    }
  return *this;
}

// MANIPULATORS

// -----------------------------------------------------------
//  MEMBER FUNCTION : clear
// -----------------------------------------------------------

void
LmpBidSet::clear()                           // remove all elements
{
  s_logger->repx(logga::adhc, "entering member function", "");
  d_bidset.clear();
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : pushString
// -----------------------------------------------------------
//  Description  : bidset parsing code for say "40.00e+06 28.00e-09 * 20.00e+06 40.00e-09"
//  Role         : loading a 'BidSet' instance using model input data
//  Techniques   : Boost.String_algo, calls 'pushBid' for actual insertion into 'd_bidset'
//  Status       : complete but not tested
//
//  Design notes
//
//      As currently coded, this member function can be called
//      multiple times.
//
//      This code could have used the Boost.Tokenizer library,
//      but 'boost::algorithm::split' from the Boost.String_algo
//      library is employed instead.
//
//      The "sCase" prefix is used to indicate a string variable.
```

```
//
//      Note that 'continue' and 'break' are supported by the
//      'Boost.Foreach' library.
//
// ----------------------------------------------------------

int                                     // number of bids loaded this time
LmpBidSet::pushString                   // parses input, uses 'pushBid' to load data
(const std::string sBidset)
{
  s_logger->repx(logga::adhc, "entering member function, strlen", sBidset.length());
  int loopCount = 0;                    // also the return value

  // defensive programming
  if ( sBidset.empty() )
    {
      s_logger->repx(logga::warn, "empty bidset string", "");
      return loopCount;
    }

#if 0 // 0 = multiple push calls supported, 1 = 'd_bidset' cleared each time

  clear();                              // member function

#endif // 0

  // split input string into bids
  std::vector<std::string> sBids;                 // split on bid separator
  const std::string bs = xeona::modelBidDelim;    // bid separator set in 'common.cc'
  boost::algorithm::split(sBids,
                          sBidset,
                          boost::algorithm::is_any_of(bs),  // bid separator
                          boost::algorithm::token_compress_off);

  // process individual bids
  std::vector<std::string> fields;      // split on price-quantity separator
  bid_type bid;                         // bid for insertion
  BOOST_FOREACH( std::string sBid, sBids )
    {
      // split individual bids into fields
      fields.clear();                   // remove all elements
      boost::algorithm::trim(sBid);     // CAUTION: 'token_compress_on' insufficient
      boost::algorithm::split(fields,
                              sBid,
                              boost::algorithm::is_any_of(" "),  // field separator
                              boost::algorithm::token_compress_on);
      // confirm structure
      const int elementCount = fields.size();
      if ( elementCount != 2 )
        {
          s_logger->repx(logga::warn, "bid field size not two", elementCount);
          std::ostringstream put;
          put << "  problematic bid string"                                        << "\n"
              << "    label              :  "         << d_label                  << "\n"
              << "    element count      :  "         << elementCount             << "\n"
              << "    trimmed bid string : " << "\"" << sBid          << "\""     << "\n";
          int loop = 0;
          BOOST_FOREACH( std::string s, fields )
            {
              put << "    element " << ++loop << "          :  " << s << "\n";
            }
          s_logger->putx(logga::xtra, put);
          s_logger->repx(logga::xtra, "abandoning bid processing", "");
          continue;                     // could also be 'break'
        }
      // attempt a string to double cast
      try
        {
          bid.first  = boost::lexical_cast<double>(fields.at(0));
          bid.second = boost::lexical_cast<double>(fields.at(1));
        }
      catch( const boost::bad_lexical_cast& eblc)
        {
          // what "bad lexical cast: source type value could not be interpreted as target"
          std::ostringstream put;
          put << "  " << "bidset string to double cast failure" << "\n"
              << "  " << "split strings: " << fields.at(0) << " " << fields.at(1) << "\n"
              << "  " << "what: " << eblc.what() << "\n";
          s_logger->putx(logga::warn, put);
          continue;
        }
```

```
      pushBid(bid);                         // member function for sorted insertion
      ++loopCount;
    }

  return loopCount;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : pushBid
// -----------------------------------------------------------

// the 'std::upper_bound' algorithm expects a sorted sequence
// -- for which a 'std::vector' container is suitable as long
// as it remains sorted

int                                     // current number of bids
LmpBidSet::pushBid
(const bid_type bid)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // range checking
  const double price = bid.second;
  if ( price < 0.0 )
    {
      s_logger->repx(logga::rankJumpy, "negative unit price encountered", price);
    }

  // active code
  d_bidset.insert                       // refer <vector>
    (std::upper_bound(d_bidset.begin(),     // refer <algorithm>
                      d_bidset.end(),
                      bid,
                      &::bid_less),         // comparison predicate, defined in this file
     bid);                                  // value to insert
  return size();                            // member function
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : negate
// -----------------------------------------------------------

void
LmpBidSet::negate()                     // multiply all unit prices by minus one
{
  d_negate = true;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : unnegate
// -----------------------------------------------------------

void
LmpBidSet::unnegate()                   // cancel any earlier 'negate' 'call'
{
  d_negate = false;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : popBig
// -----------------------------------------------------------

LmpBidSet::bid_type
LmpBidSet::popBig()                     // pop in sorted order
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( d_bidset.empty() )
    {
      s_logger->repx(logga::warn, "attempt to pop an empty bidset", size());
      return bid_type(0.0, 0.0);
    }

  // active code
  bid_type temp;
  temp = d_bidset.back();               // CAUTION: 'back' requires element to exist
  d_bidset.pop_back();                  // CAUTION: 'pop_back' doesn't return element
  if ( d_negate ) temp.second *= -1;
```

```
  return temp;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : popSmall
// ----------------------------------------------------------

LmpBidSet::bid_type
LmpBidSet::popSmall()                          // pop in sorted order
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( d_bidset.empty() )
    {
      s_logger->repx(logga::warn, "attempt to pop an empty bidset", size());
      return bid_type(0.0, 0.0);
    }

  // active code
  bid_type temp;
  temp = d_bidset.front();                     // CAUTION: 'front' requires element to exist
  d_bidset.erase(d_bidset.begin());            // CAUTION: code requires element to exist
  if ( d_negate ) temp.second *= -1;
  return temp;
}

// ACCESSORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : size
// ----------------------------------------------------------

int                                            // current number of bids
LmpBidSet::size() const
{
  const int size =  d_bidset.size();
  s_logger->repx(logga::adhc, "entering member function, size", size);
  return size;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLoQuantity
// ----------------------------------------------------------

double
LmpBidSet::getLoQuantity() const
{
  return 0.0;                                  // by definition
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getHiQuantity
// ----------------------------------------------------------

double
LmpBidSet::getHiQuantity() const
{
  double sum = 0.0;
  BOOST_FOREACH( bid_type bid, d_bidset )      // CAUTION: BOOST_FOREACH requires typedef
    {
      sum += bid.first;
    }
  return sum;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getWeightedPrice
// ----------------------------------------------------------
//  Description  : returns weighted average unit price
//  Role         : general use
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//                         sum( q_i * p_i )
//      weighted-price = ------------------  for i in 1 .. bid-count
//                            sum( q_i )
//
//      returns explicit 'nan' if 'd_bidset' is empty
```

```
//
//      returns calculated 'nan' if zero hi-quantity (see above)
//
// ------------------------------------------------------------

double
LmpBidSet::getWeightedPrice() const
{
  // empty bidset case
  if ( d_bidset.empty() )
    {
      return std::numeric_limits<double>::quiet_NaN(); // empty bidset
    }

  // active code
  double quan      = 0.0;
  double priceQuan = 0.0;
  BOOST_FOREACH( bid_type bid, d_bidset )    // CAUTION: BOOST_FOREACH requires typedef
    {
      quan      +=  bid.first;               // sum( q_i )
      priceQuan += (bid.first * bid.second); // sum( q_i * p_i )
    }
  return ( priceQuan / quan );               // can also be 'NaN'
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : stringify
// ------------------------------------------------------------

std::string                                  // 'modelBidDelim'-separated for output
LmpBidSet::stringify() const
{
  std::ostringstream buffer;
#ifndef _XUTEST
  buffer << std::scientific;
#else
  buffer << std::scientific;
  buffer << std::setprecision(2);            // useful for unit testing
#endif // _XUTEST

  const int bids = d_bidset.size();

  int loop = 0;                              // loop counter
  BOOST_FOREACH( bid_type bid, d_bidset )    // CAUTION: BOOST_FOREACH requires typedef
    {
      buffer << bid.first << " " << bid.second;
      if ( ++loop == bids ) break;
      buffer << " " << xeona::modelBidDelim << " ";    // add the intervening separator
    }

  return buffer.str();
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : summarizeAll
// ------------------------------------------------------------
//  Description  : output 'd_bidset' in human readable form
//  Role         : development purposes
//  Techniques   : 'Boost.Format'
//  Status       : complete
//
//  Design notes
//
//      Typical output
//
//          The following is typical.  The first part has
//          suppressed precision, whereas the the part in
//          brackets does not.  The currency string is now "$."
//
//              40.00e+06 W  28.00e-09 $/J  (4e+07, 2.8e-08)
//
// ------------------------------------------------------------

std::string
LmpBidSet::summarizeAll
(const int indent) const
{
  s_logger->repx(logga::adhc, "entering member function", "");

  const std::string tab(indent, ' ');
  std::ostringstream oss;
```

```
  oss << tab << "bidset label: " << d_label << "\n";
  BOOST_FOREACH( bid_type bid, d_bidset )    // CAUTION: BOOST_FOREACH requires typedef
    {
      oss << tab << "  "
          << boost::format("% 6.2fe+06 W")    % (bid.first  / 1e6)
          << " "
          << boost::format("% 6.2fe-09 $/J")  % (bid.second * 1e9)
          << "  " << "(" << bid.first << ", " << bid.second << ")"
          << "\n";
    }
  return oss.str();
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : node.cc
//  file-create-date : Tue 03-Nov-2009 12:18 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : LMP node entity / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/node.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "node.h"             // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  MEMBER FUNCTION : LmpNode
// ----------------------------------------------------------

LmpNode::LmpNode
(const std::string entityId,
 Record&           record) :
  Block(entityId, record),
  TicToc(xeona::e_auctionLmp),               // general: 'xeona::e_commitmentModes'
  d_nodalPrices(record.tieTimeseries<double>("nodal-prices"))
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~LmpNode
// ----------------------------------------------------------

LmpNode::~LmpNode()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : node01.cc
//  file-create-date : Tue 03-Nov-2009 12:19 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete LMP nodes 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/node01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "node01.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optnode.h"     // node optimization sub-problems for LMP nodes
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : Node1Inj <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : Node1Inj
// ----------------------------------------------------------

template <typename C>
Node1Inj<C>::Node1Inj
(const std::string entityId,
 Record&           record) :
  LmpNode(entityId, record),
  d_nodeCommodity(record.tieSingle<std::string>("node-commodity")),
  d_cable(Cable<C>::create
          (entityId,                          // me
           record.tieSingle<std::string>("socket-1"),
           d_nodeCommodity)),                 // common value
  d_socket(Socket<C>::create
           (entityId,                         // me
            "grid-1",                         // hardcoded socket label
            d_nodeCommodity)),                // common value
  d_ops()
```

```
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Node1Inj
// ----------------------------------------------------------

template <typename C>
Node1Inj<C>::~Node1Inj()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------

template <typename C>
const int                                   // zero
Node1Inj<C>::constrain
(const xeona::DomainMode capacityMode)      // not used
{

  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "Node1Inj");

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new LmpCab1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int cabGol = -1;                          // nonsensical value
  int socGol = -1;                          // nonsensical value

  // upload the engineering
  boost::tie(cabGol,                        // normal
             socGol)                        // bidirectional
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable ->bindOsp(d_solver, cabGol);
  d_socket->bindOsp(d_solver, socGol);

  // return zero
  return 0;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

template <typename C>
void
Node1Inj<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double injectionQuantity = -1.0;          // nonsensical value
  double nodalPrice        = -1.0;          // nonsensical value
  boost::tie(injectionQuantity, nodalPrice) = d_ops->downloadSolution();

  // store entity state information
  d_nodalPrices->at(d_step) = nodalPrice;

  // store some on-the-fly statistics
  d_dutyStats(injectionQuantity);           // functor provided by class 'Block'
  d_sizeStats();                            // functor provided by class 'Block'
}
```

```
// ----------------------------------------------------------
//  CLASS           : Node1Xit <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : Node1Xit
// ----------------------------------------------------------

template <typename C>
Node1Xit<C>::Node1Xit
(const std::string entityId,
 Record&           record) :
  LmpNode(entityId, record),
  d_nodeCommodity(record.tieSingle<std::string>("node-commodity")),
  d_cable(Cable<C>::create
          (entityId,                          // me
           record.tieSingle<std::string>("socket-1"),
           d_nodeCommodity)),                 // common value
  d_socket(Socket<C>::create
           (entityId,                         // me
            "sock-1",                         // hardcoded socket label
            d_nodeCommodity)),                // common value
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Node1Xit
// ----------------------------------------------------------

template <typename C>
Node1Xit<C>::~Node1Xit()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------

template <typename C>
const int                                     // zero
Node1Xit<C>::constrain
(const xeona::DomainMode capacityMode)        // not used
{

  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "Node1Xit");

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new LmpSoc1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int cabGol = -1;                            // nonsensical value
  int socGol = -1;                            // nonsensical value

  // upload the engineering
  boost::tie(cabGol,                          // normal
             socGol)                          // bidirectional
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable ->bindOsp(d_solver, cabGol);
  d_socket->bindOsp(d_solver, socGol);

  // return zero
  return 0;
}

// ----------------------------------------------------------
```

```
//  MEMBER FUNCTION : washup
// -------------------------------------------------------

template <typename C>
void
Node1Xit<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double exitQuantity = -1.0;               // nonsensical value
  double nodalPrice   = -1.0;               // nonsensical value
  boost::tie(exitQuantity, nodalPrice) = d_ops->downloadSolution();

  // store entity state information
  d_nodalPrices->at(d_step) = nodalPrice;

  // store some on-the-fly statistics
  d_dutyStats(exitQuantity);                // functor provided by class 'Block'
  d_sizeStats();                            // functor provided by class 'Block'
}

// -------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// -------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// class 'Commodity' typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class 'Node1Inj<>'

template Node1Inj<Oxid>::Node1Inj(const std::string, Record&);
template Node1Inj<Cert>::Node1Inj(const std::string, Record&);
template Node1Inj<Cseq>::Node1Inj(const std::string, Record&);
template Node1Inj<Elec>::Node1Inj(const std::string, Record&);
template Node1Inj<Work>::Node1Inj(const std::string, Record&);
template Node1Inj<Heat>::Node1Inj(const std::string, Record&);
template Node1Inj<Thrm>::Node1Inj(const std::string, Record&);
template Node1Inj<Fund>::Node1Inj(const std::string, Record&);

template Node1Inj<Oxid>::~Node1Inj();
template Node1Inj<Cert>::~Node1Inj();
template Node1Inj<Cseq>::~Node1Inj();
template Node1Inj<Elec>::~Node1Inj();
template Node1Inj<Work>::~Node1Inj();
template Node1Inj<Heat>::~Node1Inj();
template Node1Inj<Thrm>::~Node1Inj();
template Node1Inj<Fund>::~Node1Inj();

template const int Node1Inj<Oxid>::constrain(const xeona::DomainMode);
template const int Node1Inj<Cert>::constrain(const xeona::DomainMode);
template const int Node1Inj<Cseq>::constrain(const xeona::DomainMode);
template const int Node1Inj<Elec>::constrain(const xeona::DomainMode);
template const int Node1Inj<Work>::constrain(const xeona::DomainMode);
template const int Node1Inj<Heat>::constrain(const xeona::DomainMode);
template const int Node1Inj<Thrm>::constrain(const xeona::DomainMode);
template const int Node1Inj<Fund>::constrain(const xeona::DomainMode);

template void Node1Inj<Oxid>::washup();
template void Node1Inj<Cert>::washup();
template void Node1Inj<Cseq>::washup();
template void Node1Inj<Elec>::washup();
template void Node1Inj<Work>::washup();
template void Node1Inj<Heat>::washup();
template void Node1Inj<Thrm>::washup();
template void Node1Inj<Fund>::washup();
```

```
// class 'Node1Xit<>'

template Node1Xit<Oxid>::Node1Xit(const std::string, Record&);
template Node1Xit<Cert>::Node1Xit(const std::string, Record&);
template Node1Xit<Cseq>::Node1Xit(const std::string, Record&);
template Node1Xit<Elec>::Node1Xit(const std::string, Record&);
template Node1Xit<Work>::Node1Xit(const std::string, Record&);
template Node1Xit<Heat>::Node1Xit(const std::string, Record&);
template Node1Xit<Thrm>::Node1Xit(const std::string, Record&);
template Node1Xit<Fund>::Node1Xit(const std::string, Record&);

template Node1Xit<Oxid>::~Node1Xit();
template Node1Xit<Cert>::~Node1Xit();
template Node1Xit<Cseq>::~Node1Xit();
template Node1Xit<Elec>::~Node1Xit();
template Node1Xit<Work>::~Node1Xit();
template Node1Xit<Heat>::~Node1Xit();
template Node1Xit<Thrm>::~Node1Xit();
template Node1Xit<Fund>::~Node1Xit();

template const int Node1Xit<Oxid>::constrain(const xeona::DomainMode);
template const int Node1Xit<Cert>::constrain(const xeona::DomainMode);
template const int Node1Xit<Cseq>::constrain(const xeona::DomainMode);
template const int Node1Xit<Elec>::constrain(const xeona::DomainMode);
template const int Node1Xit<Work>::constrain(const xeona::DomainMode);
template const int Node1Xit<Heat>::constrain(const xeona::DomainMode);
template const int Node1Xit<Thrm>::constrain(const xeona::DomainMode);
template const int Node1Xit<Fund>::constrain(const xeona::DomainMode);

template void Node1Xit<Oxid>::washup();
template void Node1Xit<Cert>::washup();
template void Node1Xit<Cseq>::washup();
template void Node1Xit<Elec>::washup();
template void Node1Xit<Work>::washup();
template void Node1Xit<Heat>::washup();
template void Node1Xit<Thrm>::washup();
template void Node1Xit<Fund>::washup();

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : node02.cc
//  file-create-date : Wed 04-Nov-2009 11:12 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete LMP nodes 2 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/node02.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "node02.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optnode.h"     // node optimization sub-problems for LMP nodes
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : Node2Nul <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : Node2Nul
// ----------------------------------------------------------

template <typename C>
Node2Nul<C>::Node2Nul
(const std::string entityId,
 Record&           record) :
  LmpNode(entityId, record),
  d_nodeCommodity(record.tieSingle<std::string>("node-commodity")),
  d_cable(Cable<C>::create
          (entityId,                          // me
           record.tieSingle<std::string>("socket-1"),
           d_nodeCommodity)),                 // common value
  d_socket(Socket<C>::create
           (entityId,                         // me
            "grid-1",                         // hardcoded socket label
            d_nodeCommodity)),                // common value
  d_ops()
```

```
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Node2Nul
// ----------------------------------------------------------

template <typename C>
Node2Nul<C>::~Node2Nul()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------

template <typename C>
const int                                    // zero
Node2Nul<C>::constrain
(const xeona::DomainMode capacityMode)       // not used
{

  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "Node2Nul");

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new LmpNul2(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int cab1Gol = -1;                           // nonsensical value
  int soc1Gol = -1;                           // nonsensical value

  // upload the engineering
  boost::tie(cab1Gol,                         // bidirectional
             soc1Gol)                         // bidirectional
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable ->bindOsp(d_solver, cab1Gol);
  d_socket->bindOsp(d_solver, soc1Gol);

  // return zero
  return 0;

} // function 'constrain'

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

template <typename C>
void
Node2Nul<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double throughputQuantity = -1.0;          // nonsensical value
  double nodalPrice         = -1.0;          // nonsensical value
  boost::tie(throughputQuantity, nodalPrice) = d_ops->downloadSolution();

  // store entity state information
  d_nodalPrices->at(d_step) = nodalPrice;

  // store some on-the-fly statistics
  d_dutyStats(throughputQuantity);           // functor provided by class 'Block'
  d_sizeStats();                             // functor provided by class 'Block'
}
```

```
// ----------------------------------------------------------
//  CLASS           : Node2Inj <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : Node2Inj
// ----------------------------------------------------------

template <typename C>
Node2Inj<C>::Node2Inj
(const std::string entityId,
 Record&           record) :
  LmpNode(entityId, record),
  d_nodeCommodity(record.tieSingle<std::string>("node-commodity")),
  d_cable1(Cable<C>::create
           (entityId,                          // me
            record.tieSingle<std::string>("socket-1"),
            d_nodeCommodity)),                 // common value
  d_cable2(Cable<C>::create
           (entityId,                          // me
            record.tieSingle<std::string>("socket-2"),
            d_nodeCommodity)),                 // common value
  d_socket(Socket<C>::create
           (entityId,                         // me
            "grid-1",                         // hardcoded socket label
            d_nodeCommodity)),                // common value
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Node2Inj
// ----------------------------------------------------------

template <typename C>
Node2Inj<C>::~Node2Inj()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------

template <typename C>
const int                                      // zero
Node2Inj<C>::constrain
(const xeona::DomainMode capacityMode)         // not used
{

  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "Node2Inj");

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new LmpCab2(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int cab1Gol = -1;                            // nonsensical value
  int cab2Gol = -1;                            // nonsensical value
  int soc1Gol = -1;                            // nonsensical value

  // upload the engineering
  boost::tie(cab1Gol,                          // normal
             cab2Gol,                          // bidirectional
             soc1Gol)                          // bidirectional
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable1->bindOsp(d_solver, cab1Gol);
```

```
  d_cable2->bindOsp(d_solver, cab2Gol);
  d_socket->bindOsp(d_solver, soc1Gol);

  // return zero
  return 0;

} // function 'constrain'

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

template <typename C>
void
Node2Inj<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double injectionQuantity = -1.0;           // nonsensical value
  double nodalPrice        = -1.0;           // nonsensical value
  boost::tie(injectionQuantity, nodalPrice) = d_ops->downloadSolution();

  // store entity state information
  d_nodalPrices->at(d_step) = nodalPrice;

  // store some on-the-fly statistics
  d_dutyStats(injectionQuantity);            // functor provided by class 'Block'
  d_sizeStats();                             // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  CLASS           : Node2Xit <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : Node2Xit
// ----------------------------------------------------------

template <typename C>
Node2Xit<C>::Node2Xit
(const std::string entityId,
 Record&           record) :
  LmpNode(entityId, record),
  d_nodeCommodity(record.tieSingle<std::string>("node-commodity")),
  d_cable(Cable<C>::create
          (entityId,                         // me
           record.tieSingle<std::string>("socket-1"),
           d_nodeCommodity)),                // common value
  d_socket1(Socket<C>::create
            (entityId,                       // me
             "sock-1",                       // hardcoded socket label
             d_nodeCommodity)),              // common value
  d_socket2(Socket<C>::create
            (entityId,                       // me
             "grid-1",                       // hardcoded socket label
             d_nodeCommodity)),              // common value
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Node2Xit
// ----------------------------------------------------------

template <typename C>
Node2Xit<C>::~Node2Xit()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
```

```
template <typename C>
const int                                       // zero
Node2Xit<C>::constrain
(const xeona::DomainMode capacityMode)          // not used
{

  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "Node2Xit");

  // create and fill a label object
  const std::string junctionId = getIdentifier();
  Label lab(junctionId);

  // recreate new operations OSP of the required type
  d_ops.reset(new LmpSoc2(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int cab1Gol = -1;                              // nonsensical value
  int soc1Gol = -1;                              // nonsensical value
  int soc2Gol = -1;                              // nonsensical value

  // upload the engineering
  boost::tie(cab1Gol,                            // bidirectional
             soc1Gol,                            // normal
             soc2Gol)                            // bidirectional
    = d_ops->uploadEngineering();

  // bind global cols to the relevant interfaces
  d_cable  ->bindOsp(d_solver, cab1Gol);
  d_socket1->bindOsp(d_solver, soc1Gol);
  d_socket2->bindOsp(d_solver, soc2Gol);

  // return zero
  return 0;

} // function 'constrain'

// ------------------------------------------------------------
//  MEMBER FUNCTION : washup
// ------------------------------------------------------------

template <typename C>
void
Node2Xit<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double exitQuantity = -1.0;                    // nonsensical value
  double nodalPrice   = -1.0;                    // nonsensical value
  boost::tie(exitQuantity, nodalPrice) = d_ops->downloadSolution();

  // store entity state information
  d_nodalPrices->at(d_step) = nodalPrice;

  // store some on-the-fly statistics
  d_dutyStats(exitQuantity);                     // functor provided by class 'Block'
  d_sizeStats();                                 // functor provided by class 'Block'
}

// ------------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ------------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// class 'Commodity' typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
```

```
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class 'Node2Nul<>'

template Node2Nul<Oxid>::Node2Nul(const std::string, Record&);
template Node2Nul<Cert>::Node2Nul(const std::string, Record&);
template Node2Nul<Cseq>::Node2Nul(const std::string, Record&);
template Node2Nul<Elec>::Node2Nul(const std::string, Record&);
template Node2Nul<Work>::Node2Nul(const std::string, Record&);
template Node2Nul<Heat>::Node2Nul(const std::string, Record&);
template Node2Nul<Thrm>::Node2Nul(const std::string, Record&);
template Node2Nul<Fund>::Node2Nul(const std::string, Record&);

template Node2Nul<Oxid>::~Node2Nul();
template Node2Nul<Cert>::~Node2Nul();
template Node2Nul<Cseq>::~Node2Nul();
template Node2Nul<Elec>::~Node2Nul();
template Node2Nul<Work>::~Node2Nul();
template Node2Nul<Heat>::~Node2Nul();
template Node2Nul<Thrm>::~Node2Nul();
template Node2Nul<Fund>::~Node2Nul();

template const int Node2Nul<Oxid>::constrain(const xeona::DomainMode);
template const int Node2Nul<Cert>::constrain(const xeona::DomainMode);
template const int Node2Nul<Cseq>::constrain(const xeona::DomainMode);
template const int Node2Nul<Elec>::constrain(const xeona::DomainMode);
template const int Node2Nul<Work>::constrain(const xeona::DomainMode);
template const int Node2Nul<Heat>::constrain(const xeona::DomainMode);
template const int Node2Nul<Thrm>::constrain(const xeona::DomainMode);
template const int Node2Nul<Fund>::constrain(const xeona::DomainMode);

template void Node2Nul<Oxid>::washup();
template void Node2Nul<Cert>::washup();
template void Node2Nul<Cseq>::washup();
template void Node2Nul<Elec>::washup();
template void Node2Nul<Work>::washup();
template void Node2Nul<Heat>::washup();
template void Node2Nul<Thrm>::washup();
template void Node2Nul<Fund>::washup();

// class 'Node2Inj<>'

template Node2Inj<Oxid>::Node2Inj(const std::string, Record&);
template Node2Inj<Cert>::Node2Inj(const std::string, Record&);
template Node2Inj<Cseq>::Node2Inj(const std::string, Record&);
template Node2Inj<Elec>::Node2Inj(const std::string, Record&);
template Node2Inj<Work>::Node2Inj(const std::string, Record&);
template Node2Inj<Heat>::Node2Inj(const std::string, Record&);
template Node2Inj<Thrm>::Node2Inj(const std::string, Record&);
template Node2Inj<Fund>::Node2Inj(const std::string, Record&);

template Node2Inj<Oxid>::~Node2Inj();
template Node2Inj<Cert>::~Node2Inj();
template Node2Inj<Cseq>::~Node2Inj();
template Node2Inj<Elec>::~Node2Inj();
template Node2Inj<Work>::~Node2Inj();
template Node2Inj<Heat>::~Node2Inj();
template Node2Inj<Thrm>::~Node2Inj();
template Node2Inj<Fund>::~Node2Inj();

template const int Node2Inj<Oxid>::constrain(const xeona::DomainMode);
template const int Node2Inj<Cert>::constrain(const xeona::DomainMode);
template const int Node2Inj<Cseq>::constrain(const xeona::DomainMode);
template const int Node2Inj<Elec>::constrain(const xeona::DomainMode);
template const int Node2Inj<Work>::constrain(const xeona::DomainMode);
template const int Node2Inj<Heat>::constrain(const xeona::DomainMode);
template const int Node2Inj<Thrm>::constrain(const xeona::DomainMode);
template const int Node2Inj<Fund>::constrain(const xeona::DomainMode);

template void Node2Inj<Oxid>::washup();
template void Node2Inj<Cert>::washup();
template void Node2Inj<Cseq>::washup();
template void Node2Inj<Elec>::washup();
template void Node2Inj<Work>::washup();
template void Node2Inj<Heat>::washup();
template void Node2Inj<Thrm>::washup();
template void Node2Inj<Fund>::washup();

// class 'Node2Xit<>'
```

```
template Node2Xit<Oxid>::Node2Xit(const std::string, Record&);
template Node2Xit<Cert>::Node2Xit(const std::string, Record&);
template Node2Xit<Cseq>::Node2Xit(const std::string, Record&);
template Node2Xit<Elec>::Node2Xit(const std::string, Record&);
template Node2Xit<Work>::Node2Xit(const std::string, Record&);
template Node2Xit<Heat>::Node2Xit(const std::string, Record&);
template Node2Xit<Thrm>::Node2Xit(const std::string, Record&);
template Node2Xit<Fund>::Node2Xit(const std::string, Record&);

template Node2Xit<Oxid>::~Node2Xit();
template Node2Xit<Cert>::~Node2Xit();
template Node2Xit<Cseq>::~Node2Xit();
template Node2Xit<Elec>::~Node2Xit();
template Node2Xit<Work>::~Node2Xit();
template Node2Xit<Heat>::~Node2Xit();
template Node2Xit<Thrm>::~Node2Xit();
template Node2Xit<Fund>::~Node2Xit();

template const int Node2Xit<Oxid>::constrain(const xeona::DomainMode);
template const int Node2Xit<Cert>::constrain(const xeona::DomainMode);
template const int Node2Xit<Cseq>::constrain(const xeona::DomainMode);
template const int Node2Xit<Elec>::constrain(const xeona::DomainMode);
template const int Node2Xit<Work>::constrain(const xeona::DomainMode);
template const int Node2Xit<Heat>::constrain(const xeona::DomainMode);
template const int Node2Xit<Thrm>::constrain(const xeona::DomainMode);
template const int Node2Xit<Fund>::constrain(const xeona::DomainMode);

template void Node2Xit<Oxid>::washup();
template void Node2Xit<Cert>::washup();
template void Node2Xit<Cseq>::washup();
template void Node2Xit<Elec>::washup();
template void Node2Xit<Work>::washup();
template void Node2Xit<Heat>::washup();
template void Node2Xit<Thrm>::washup();
template void Node2Xit<Fund>::washup();

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optctl.cc
//  file-create-date : Fri 17-Oct-2008 14:36 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : control OSPs for asset operators / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optctl.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "optctl.h"           // companion header for this file (place first)

#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
#include "../c/util2.h"       // free functions which offer general utilities 2
#include "../c/label.h"       // helper class to format solver labels
#include "../c/costs.h"       // cost sets and support
#include "../b/lmpbid.h"      // LMP auction bidset

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>             // printf style formatting
#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  CODE

// ----------------------------------------------------------
//  CLASS           : CtlFirstFeasible_A
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtlFirstFeasible_A
// ----------------------------------------------------------

CtlFirstFeasible_A::CtlFirstFeasible_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  ControlOsp(solver, commitmentMode, xeona::e_adminFirst, "ctl-firstfeas-a")
{
```

```
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CtlFirstFeasible_A
// ----------------------------------------------------------

CtlFirstFeasible_A::~CtlFirstFeasible_A()
{
  s_logger->repx(logga::xtra, "destructor call", "");
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadNullControl
// ----------------------------------------------------------
//  Description  : fill the solver using intermediate calls
//  Role         : host usage
//  Techniques   : (nothing special)
//  Status       : first-pass complete and tested
// ----------------------------------------------------------

void
CtlFirstFeasible_A::uploadNullControl()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // ACTIVE CODE

#if 0 // 0 = no active code, 1 = some active code

  // process label
  Label lab(d_label);

  // col calls
  pushObj(lab.str("null"));

#endif // 0

}

// SPECIALIZED PUSH CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : pushObj
// ----------------------------------------------------------

const int
CtlFirstFeasible_A::pushObj
(const std::string tag)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "CtlFirstFeasible");

  // ACTIVE CODE

  const double objValue = 0.0;

  // store the objective value
  d_nulObjs.push_back(objValue);

  // load the solver
  ++d_colCount;
  d_solver->loadObj(globalcol(d_colCount), objValue, tag);
  return d_colCount;
}

// ----------------------------------------------------------
//  CLASS           : CtlMeritOrder_A
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtlMeritOrder_A
// ----------------------------------------------------------

CtlMeritOrder_A::CtlMeritOrder_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
```

```
  ControlOsp(solver, commitmentMode, xeona::e_adminMerit, "ctl-merit-a"),
  d_unitLimit(0),                           // zero means no limit
  d_rankings()                              // empty vector
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CtlMeritOrder_A
// ----------------------------------------------------------

CtlMeritOrder_A::~CtlMeritOrder_A()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadRank
// ----------------------------------------------------------
//  Description  : fill the solver using intermediate calls
//  Role         : host usage
//  Techniques   : sorted vectors
//  Status       : first-pass complete but not tested but looking okay
//
//  Design notes
//
//      Terminology
//
//          This function implements "prescribed merit order".
//
//          The qualifier "prescribed" is added to
//          differentiate it from terminology used to describe
//          (somewhat optimistically) supply curves in
//          wholesale electricity markets (in Germany at
//          least).
//
//      Merit order ranking
//
//          The following assumes a 'shift' of unity.  Otherwise
//          see comments in the code.  (CAUTION: this shift is
//          different from the GLPK objective "shift" or constant
//          term)
//
//          The merit order ranking must be an integer between
//          unity and 1024 inclusive and the ordering is from
//          best to worst.  This means that a rank 1 asset will
//          always be fully dispatched in preference to any
//          other asset -- and so on down the merit order
//          chain.
//
//          Rankings above 20 are not recommended because
//          optimization scaling issues may occur.  Twenty
//          merit order assets means that the objective
//          function coefficients will range from 1 to 524288
//          (it might be useful to change the 'shift' from 1 to
//          5 say to improve the spread, relative to
//          coefficients from other sources).
//
//    Upper bound 'up'
//
//         The variable 'up' is set so that its associated
//         constraint will (hopefully) never bind.  It is
//         therefore like the "big M" concept in mathematical
//         programming (although that concept had several
//         interpretations).
//
//    Sorted vector 'd_rankings"
//
//          The individual ranks are held in the sorted vector
//          'd_rankings'.  The sorting code comes from Karlsson
//          (2006 ch12).
//
// ----------------------------------------------------------

CtlMeritOrder_A::index_type
CtlMeritOrder_A::uploadRank
(const int    rank,               // a merit order from the series { 1, 2, 3, ..., 1024 }
 const double ceilingDuty)        // used to set a "big M"-style structural coefficient
```

```
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // PRELIMINARY CHECKS

  // defensive programming
  if ( rank < 1 )
    {
      s_logger->repx(logga::warn, "rank value must be one or greater", rank);
      return 0;
    }
  else if ( rank > 1024 )                    // assuming (see below) 'shift' >= 1
    {
      s_logger->repx(logga::warn, "rank value exceeds 1024", rank);
      return 0;
    }
  else if ( rank > 20 )
    {
      s_logger->repx(logga::info, "rank value above 20 inadvisable", rank);
    }

  if ( ceilingDuty < 0.0 )
    {
      s_logger->repx(logga::warn, "negative ceilingDuty not legal", ceilingDuty);
      return 0;
    }

  // check to see if the 'rank' has already been used
  //
  // 'std::binary_search' is a non-modifying STL searching
  // algorithm for locating certain values in sorted ranges
  //
  // note also that 'd_rankings' is declared in this class and
  // is not inherited

  if ( std::binary_search                                              // refer <algorithm>
       (d_rankings.begin(), d_rankings.end(), rank) )
    {
      // not entirely sure how this could happen but the check remains useful
      s_logger->repx(logga::warn, "given rank previously registered", rank);
      return 0;
    }

  // save the current 'rank' at the appropriate location in 'd_ranks'
  //
  // the 'std::lower_bound' algorithm gives this location, but
  // also expects a sorted sequence, for which a 'std::vector'
  // container is suitable -- 'd_rankings' is sorted by virtue of
  // the previous code

  d_rankings.insert                                                    // refer <vector>
    (std::lower_bound(d_rankings.begin(), d_rankings.end(), rank),     // refer <algorithm>
     rank);                                                          // value to insert

  // ACTIVE CODE

  // create and fill a label object
  Label lab(d_label);
  lab << boost::format("upload-rank-%02d") % rank;          // 'boost::format' streamable

  // convert rank value to mapped binary series
  //
  // for instance, if shift = 1, then { 1, 2, 3, 4 } maps to { 1, 2, 4, 8 }
  const int shift     = 1;                                  // see design notes
  const double objVal = std::pow(2.0, rank - shift);        // refer <cmath>

  // col calls
  const int col = pushObj(objVal, lab.str("dispatch"));     // add dispatch binary var
  markBinary(col);                                          // mark as such
  const int dutyCol = pushObj(0.0, lab.str("dutycol"));     // add duty col

  // first bump the 'ceilingDuty' upwards a bit to make sure it
  // never binds -- noting that duty is in [W] and is unlikely
  // to be less than 1000 (but we cannot guarantee that) and
  // also zero stays at zero
  const double up = 2.0 * ceil(ceilingDuty);

  // constraint calls
  const int row = pushRhs(0.0, svif::L, lab.str("constraint"));  // set up constraint
  pushCof(row, dutyCol, +1.0);               // couple to duty variable
```

```
  pushCof(row, col,     -up);               // add binary variable coefficient

  // housekeeping
  d_dutyCol = dutyCol;                     // store duty col for possible later use
  return globalcol(d_dutyCol);             // return duty gol for coupling purposes
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadUnitLimit
// ----------------------------------------------------------

void
CtlMeritOrder_A::uploadUnitLimit
(const int unitLimit)
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function, unitlimit", unitLimit);

  // PRELIMINARY CHECKS

  if ( d_dutyCol == 0 )
    {
      s_logger->repx(logga::warn, "must follow uploadRank call", d_dutyCol);
      return;
    }

  // ACTIVE CODE

  d_unitLimit       = unitLimit;
  // TOFIX: merit order control: add unit limit constraint in due course
  // const int dutyCol = d_dutyCol;
  s_logger->repx(logga::warn, "unsupported feature, ignoring call", "");
}

// ----------------------------------------------------------
//  CLASS           : CtlLmpBid_A
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtlLmpBid_A
// ----------------------------------------------------------

CtlLmpBid_A::CtlLmpBid_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  ControlOsp(solver, commitmentMode, xeona::e_auctionLmp, "ctl-lmpbid-a")
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call, commitment mode", commitmentMode);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CtlLmpBid_A
// ----------------------------------------------------------

CtlLmpBid_A::~CtlLmpBid_A()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadBidSet
// ----------------------------------------------------------
//  Description  : uploads an LMP (nodal pricing) bidset
//  Role         : host usage
//  Techniques   : class 'LmpBidSet' for bidsets, includes friendship
//  Status       : first-pass complete
// ----------------------------------------------------------

CtlLmpBid_A::index_type                    // duty control gol
CtlLmpBid_A::uploadBidSet
(const shared_ptr<LmpBidSet> bidset)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // ACTIVE CODE

  // CAUTION: first make a copy of 'bidset' because the pop calls
```

```
  // later on destroy the bidset -- whilst noting that class
  // 'CtlLmpBid_A' is a friend of class 'LmpBidSet'

  shared_ptr<LmpBidSet> tempBidset(new LmpBidSet(*bidset));

  // declare some administration counters
  int bidCount = 0;                          // bids processed

  // create and initialize a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // first create a bidset balance
  const int setCol = pushObj(0.0, zeroSpecCosts, lab.str("bidset"));
  const int setRow = pushRhs(0.0, svif::E, lab.str("bidset-bal"));
  d_cofCount       = pushCof(setRow, setCol, -1.0);

  // then loop thru the bids
  while ( tempBidset->size() )
    {
      // update bid count
      ++bidCount;

      // set up label
      lab << boost::format("bid-%02d") %  bidCount;

      // grab biggest bid, meaning current highest unit price
      const std::pair<double, double> bid = tempBidset->popBig();
      const double deltaQuantity = bid.first;
      const double unitPrice     = bid.second;

      // additional reporting as appropriate
      // YEEK 12 CODE (set by '--yeek')
      if ( xeona::yeek == 12 || xeona::yeek == 1 )
        {
          std::ostringstream put;
          put << "  bid details"                                    << "\n"
              << "    bid number     : "
              << std::setw(2) << std::setfill('0') << bidCount      << "\n"
              << "    delta quantity : " << deltaQuantity           << "\n"
              << "    unit price     : " << unitPrice               << "\n";
          s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
          s_logger->putx(logga::dbug, put);
        }

      // define blockette bounds
      const double lowerBnd = 0;
      const double upperBnd = deltaQuantity;

      // set up blockette variables, just the "out" in this case
      const int bidCol   = pushObj(unitPrice, zeroSpecCosts, lab.str(""));  // "" is okay

      // add blockette variable to common balance
      d_cofCount         = pushCof(setRow, bidCol, +1.0);

      // add constraint for lower and upper bounds
      const int lowerRow = pushRhs(lowerBnd, svif::G, lab.str("lo-band"));
      d_cofCount         = pushCof(lowerRow, bidCol, +1.0);
      const int upperRow = pushRhs(upperBnd, svif::L, lab.str("hi-band"));
      d_cofCount         = pushCof(upperRow, bidCol, +1.0);

      // NOTE: blockette input/output relationships are not required

      lab.trim(1);                           // remove the last "bid-00" labelette
    }

  // completion reporting
  s_logger->repx(logga::adhc, "bid processing complete, count", bidCount);

  // housekeeping
  d_dutyCol = setCol;
  return globalcol(d_dutyCol);

} // function 'CtlLmpBid_A::uploadBidSet'

// ------------------------------------------------------------
//  CLASS           : CtlLeastCost_A
// ------------------------------------------------------------
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : CtlLeastCost_A
// ----------------------------------------------------------

CtlLeastCost_A::CtlLeastCost_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  ControlOsp(solver, commitmentMode, xeona::e_shortrunModes, "ctl-leastcost-a")
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CtlLeastCost_A
// ----------------------------------------------------------

CtlLeastCost_A::~CtlLeastCost_A()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadShortrunCosts
// ----------------------------------------------------------
//  Description  : upload standing costs (duty and size independent costs)
//  Role         : host usage
//  Techniques   : increments objective "shift" term
//  Status       : first-pass complete
//
//  Design notes
//
//      This call loads the short-run non-duty costs of the
//      asset operator.
//
// ----------------------------------------------------------

void
CtlLeastCost_A::uploadShortrunCosts
(const CostSet& shiftCosts)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // ACTIVE CODE

  // push increment "shift" term
  pushIncShift(shiftCosts);                 // no label required or used
}

// ----------------------------------------------------------
//  CLASS           : CtlQuan_A
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtlQuan_A
// ----------------------------------------------------------

CtlQuan_A::CtlQuan_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  ControlOsp(solver, commitmentMode, xeona::e_commitmentModes, "ctl-quan-a"),
  d_cols()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CtlQuan_A
// ----------------------------------------------------------

CtlQuan_A::~CtlQuan_A()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadControl
// ----------------------------------------------------------
//  Description  : uploads demand quantity
```

```
//  Role         : host usage
//  Techniques   : (nothing special)
//  Status       : more-or-less complete
// ----------------------------------------------------------

CtlQuan_A::index_type                        // duty control gol
CtlQuan_A::uploadControl
(const double demand)                        // demanded quantity
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);                        // entity identifier

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // INTEGRITY CHECKS

  // very basic range checking -- with negative demand being
  // considered invalid
  if ( demand < 0.0 )
    {
      s_logger->repx(logga::rankJumpy,
                     "negative demand encountered",
                     demand);
    }

  // EXPOSED VARIABLE

  // create one variable
  const int ctlCol  = pushObj(zeroSpecCosts, lab.str("control"));

  // load the local index into a 1-tuple
  d_cols            = boost::make_tuple(ctlCol);

  // DEMAND EQUALITY

  // create an input balance and add a coefficient
  const int quanRow = pushRhs(demand, svif::E, lab.str("demand"));
  d_cofCount        = pushCof(quanRow, ctlCol, +1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optgate.cc
//  file-create-date : Wed 25-Mar-2009 21:08 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : various OSPs for gateways / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optgate.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "optgate.h"          // companion header for this file (place first)

#include "../c/util2.h"       // free functions which offer general utilities 2
#include "../c/label.h"       // helper class to format solver labels
#include "../b/optops.h"      // operate optimization sub-problems for hard assets
#include "../b/optctl.h"      // control optimization sub-problems for asset operators
#include "../b/bandtaf.h"     // banded tariff set and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting

//  CODE

// NOTE: this simple way of swapping OSP implementations, the
// "_X" postfix needs changing in this one place only -- note
// also that an implementation change is required to honor the
// same function names and signatures, but not necessarily the
// behavior at large.

// ------------------------------------------------------------
//  CLASS           : QanTechCapacity_A
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : QanTechCapacity_A
// ------------------------------------------------------------

QanTechCapacity_A::QanTechCapacity_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string          tag) :
  QuantityOsp(solver, commitmentMode, "qan-tech-cap-a"),
  d_cols()
{
```

```
  s_logger->repx(logga::xtra, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~QanTechCapacity_A
// ----------------------------------------------------------

QanTechCapacity_A::~QanTechCapacity_A()
{
  s_logger->repx(logga::xtra, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadCapacity
// ----------------------------------------------------------
//  Description  : capacity characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : more-or-less complete
// ----------------------------------------------------------

QanTechCapacity_A::index_type
QanTechCapacity_A::uploadCapacity            // wrapper
(const std::pair<double, double> capacity)
{
  return uploadCapacity(capacity.first, capacity.second);
}

QanTechCapacity_A::index_type
QanTechCapacity_A::uploadCapacity
(const double loCapacity,
 const double hiCapacity)
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // INTEGRITY CHECKS

  // very basic integrity checking
  if ( loCapacity > hiCapacity ||
       loCapacity < 0.0 )
    {
      std::ostringstream oss;
      oss << loCapacity << " : " << hiCapacity;
      s_logger->repx(logga::warn, "aphysical capacities, lo : hi", oss.str());
    }

  // EXPOSED VARIABLES

  // create an exposed variable -- the zero specific cost is, of
  // course, fine
  const int capCol = pushObj(zeroSpecCosts, lab.str("capacity"));

  // load the local col index into a 1-tuple
  d_cols = boost::make_tuple(capCol);

  // CAPACITY BALANCES

  // create the capacity balances
  const int loRow  = pushRhs(loCapacity, svif::G, lab.str("lo-bal"));
  const int hiRow  = pushRhs(hiCapacity, svif::L, lab.str("hi-bal"));

  // add the exposed variable
  d_cofCount       = pushCof(loRow, capCol, +1.0);
  d_cofCount       = pushCof(hiRow, capCol, +1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'QanTechCapacity_A::uploadCapacity'
```

```
// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// -----------------------------------------------------------
//  Description  : solution recovery
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : more-or-less complete
// -----------------------------------------------------------

QanTechCapacity_A::results_type
QanTechCapacity_A::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // identify transmission results
  const double duty = downloadVar(d_cols.get<0>());

  // return
  return boost::make_tuple(duty);

} // function 'QanTechCapacity_A::downloadSolution'

// -----------------------------------------------------------
//  CLASS           : OfrTariffSet_A
// -----------------------------------------------------------

// -----------------------------------------------------------
//  MEMBER FUNCTION : OfrTariffSet_A
// -----------------------------------------------------------

OfrTariffSet_A::OfrTariffSet_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string          tag) :
  OfferOsp(solver, commitmentMode, "ofr-tariffset-a"),
  d_cols(),
  d_tariffset(),
  d_capacity()
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

// UPLOAD CALLS

// -----------------------------------------------------------
//  MEMBER FUNCTION : uploadTariffSet
// -----------------------------------------------------------
//  Description  : fill the solver using intermediate calls
//  Role         : host usage
//  Techniques   : sorted vectors
//  Status       : mostly complete, the non-convex (price reducing) tariffs need testing
//
//  Design notes
//
//      The ORDER OF INSERTION for a 'BandedTariffSet' instance
//      is significant -- in contrast to an LMP bidset in which
//      the bids are sorted on load.  This function must
//      therefore respect the order of insertion.
//
//  Piecewise linearization
//
//      The main background comes from a posting by GLPK author
//      Andrew Makhorin in 2007.  See 'glpk-sos2_02.pdf' (or
//      latter) for a typeset version, available from:
//
//        http://winglpk.sourceforge.net/media/glpk-sos2_02.pdf
//
//      Similar material is contained in my (Robbie Morrison)
//      thesis write-up.
//
//      For additional material, see Croxton etal (2003) and Keha
//      etal (2004).
//
//  References
//
//      Croxton, Keely L, Bernard Gendron, and Thomas L Magnanti.
//        2003.  A comparison of mixed-integer programming models
//        for non-convex piecewise linear cost minimization
//        problems.  Management Science.  v49 pp1268-1273.
//
```

```
//      Keha, Ahmet B, de Farias, Ismael R, Jr, and Nemhauser,
//        George L.  2004.  Models for representing piecewise
//        linear cost functions.  Operations Research Letters
//        vol32 iss1 p44-48.  ISSN 0167-6377.  doi:
//        10.1016/S0167-6377(03)00059-2.
//
// -----------------------------------------------------

OfrTariffSet_A::index_type
OfrTariffSet_A::uploadTariffSet              // forget about the fixed charge for now
(const shared_ptr<BandedTariffSet> tariffset,
 const double                      capacity)
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  Label colLabel(d_label);                   // 'OptimSubProb' data member
  Label rowLabel(d_label);                   // 'OptimSubProb' data member

  int cofCount = 0;                          // not actually used for much

  // PRELIMINARY CHECKS

  if ( tariffset->empty() )
    {
      s_logger->repx(logga::dbug, "empty tariff set submitted", tariffset->getLabel());
      return boost::make_tuple(0);
    }
  if ( capacity < 0.0 )
    {
      s_logger->repx(logga::warn, "negative capacity supplied", capacity);
    }

  // CODE

  // save tariff set and capacity for use in 'downloadSolution'
  d_tariffset = tariffset;
  d_capacity  = capacity;

  // load up OSP if using short-run financial cost minimization,
  // else create an empty problem
  switch ( d_commitmentMode )
    {
    case xeona::e_shortrunFin:               // short-run financial cost minimization
      {
        // ---------------------------------
        //  short-run financial commitment
        // ---------------------------------

        // process labels
        colLabel << tariffset->getLabel();
        rowLabel << tariffset->getLabel();

        // set up sale variable and constraint
        const int saleCol = pushObj(0.0, colLabel.str("sale"));
        const int saleRow = pushRhs(0.0, svif::E, rowLabel.str("sale-constraint"));
        cofCount          = pushCof(saleRow, saleCol, -1.0);

        // load the local index into a 1-tuple
        d_cols = boost::make_tuple(saleCol);

        // process the fixed charge
        const double specFixed = tariffset->getSpecificFixedCharge();
        if ( specFixed != 0.0 )
          {
            pushIncShift(specFixed * capacity);   // incremental
          }

        // process the bidsets based on tariff set convexity
        switch ( tariffset->isConvex() )
          {
          case true:                           // convex tariff set, linear programme
            {
              // ---------------------------------
              //  convex tariffs
              // ---------------------------------

              // convexity reporting
              s_logger->repx(logga::adhc, "convex tariffset", "");
```

```
    // loop the tariffs
    int loop = 1;
    while ( ! tariffset->empty() )
      {
        // process labels
        colLabel << boost::format("tariff-%d") % loop;
        rowLabel << boost::format("tariff-%d") % loop;

        // grab tariff, working left to right on price curve
        const tariff_type tariff = tariffset->popFirst();  // oldest current

        // tariff extensity call (band, price)
        const int col = pushObj(tariff.second, colLabel.str(""));

        // tariff constraint calls
        const int row = pushRhs(tariff.first, svif::L, rowLabel.str(""));
        cofCount      = pushCof(row, col, +1.0);

        // bind tariff to sale
        cofCount      = pushCof(saleRow, col, +1.0);

        // housekeeping
        ++loop;
      }
  }
  break;

case false:                          // non-convex tariff set, need binary vars
  {
    // ---------------------------------
    //  non-convex tariffs
    // ---------------------------------

    // TOFIX: 12-Oct-2010: buggy for non-convex tariffsets

    // convexity reporting
    s_logger->repx(logga::adhc, "non-convex tariffset", "");

    // first up, equation THREE: represented by 'saleRow'

    // define the segment endpoints: left and rite (right)
    double xleft = 0.0;
    double yleft = 0.0;
    double xrite = 0.0;            // CAUTION: must be zero
    double yrite = 0.0;            // CAUTION: must be zero

    // create equation ONE
    const int oneSegRow = pushRhs(1.0, svif::E, rowLabel.str("one-segment"));

    // loop the tariffs
    int loop = 1;
    while ( ! tariffset->empty() )
      {
        // process labels
        colLabel << boost::format("tariff-%d") % loop;
        rowLabel << boost::format("tariff-%d") % loop;

        // grab tariff, working left to right on price curve
        const tariff_type tariff = tariffset->popFirst();  // oldest current
        const double band        = tariff.first;  // band
        const double price       = tariff.second; // unit price

        // calculate the segment endpoints -- first
        // swap right to left and then add the increments
        xleft = xrite;
        yleft = yrite;
        xrite = xleft + band;
        yrite = yleft + band * price;

        // create two block local variables, trailed by "Col"
        //   "b" is a binary variable with determine which tariff is active
        //   "s" is the segment variable

        // this code also enacts equation FOUR: yleft * b + yband * s
        const int bCol = pushObj(yleft        , colLabel.str("binary"));
        const int sCol = pushObj(yrite - yleft, colLabel.str("segment"));
        markBinary(bCol);          // returns same col for convenience

        // update constraint equation ONE: sum b = 1
        cofCount       = pushCof(oneSegRow, bCol, 1.0);
```

```
                  // make new constraint equation TWO: 0 <= s <= b
                  // note default global non-negativity condition for structural variables
                  const int sRow = pushRhs(0.0, svif::G, rowLabel.str("s-equation"));
                  cofCount       = pushCof(sRow, sCol,  1.0);
                  cofCount       = pushCof(sRow, bCol, -1.0);

                  // update constraint equation THREE:  xleft * b + xband * s
                  cofCount       = pushCof(saleRow, bCol, xleft        );
                  cofCount       = pushCof(saleRow, sCol, xleft - xrite);

                  // housekeeping
                  ++loop;
                }

            }
            break;

          } // inner switch on 'isConvex' boolean

        // GLOBALIZE AND RETURN
        // note that external usage requires global indexing

        return boost::make_tuple(globalcol(saleCol));
      }

    case xeona::e_shortrunGhg:               // non-financial
    case xeona::e_shortrunNox:
    case xeona::e_shortrunDep:
    case xeona::e_shortrunLuc:
    case xeona::e_auctionLmp:
    case xeona::e_adminMerit:
    case xeona::e_adminFirst:
      {
        // ---------------------------------
        //  all other commitment modes
        // ---------------------------------

        // process labels
        colLabel << tariffset->getLabel();

        // set up sale variable and constraint
        const int saleCol = pushObj(0.0, colLabel.str("sale"));

        // GLOBALIZE AND RETURN
        // note that external usage requires global indexing

        return boost::make_tuple(globalcol(saleCol));
      }

    default:
      std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
      return boost::make_tuple(0);

    } // outer switch on 'xeona::DomainMode' enum

  // should never get here

} // function 'OfrTariffSet_A::uploadTariffSet'

// ------------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ------------------------------------------------------------
//  Description  :
//  Role         :
//  Techniques   : 'BandedTariffSet::interpretSale'
//  Status       : incomplete
//
//  Design notes
//
//      Most of the hard work is done buy
//      'BandedTariffSet::interpretSale'.
//
//      Note to that the capacity information given and utilized
//      in 'uploadTariffSet' is also needed here.
//
// ------------------------------------------------------------

OfrTariffSet_A::results_type
OfrTariffSet_A::downloadSolution
(const int interval) const
```

```
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // identify contract results
  const double size = d_capacity;
  const double sale = downloadVar(d_cols.get<0>());

  // additional reporting as appropriate
  // YEEK 32 CODE (set by '--yeek')
  if ( xeona::yeek == 32 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  OSP information"           << "\n"
          << "    recorded"                << "\n"
          << "        size : " << size     << "\n"
          << "        sale : " << sale     << "\n"
          << "     interval: " << interval << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // return (direct out is okay)
  return d_tariffset->interpretSale(size, sale, interval);

} // function 'OfrTariffSet_A::downloadSolution'

// ------------------------------------------------------------
//  CLASS           : QanObligToSupply_A
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : QanObligToSupply_A
// ------------------------------------------------------------

QanObligToSupply_A::QanObligToSupply_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string          tag) :
  QuantityOsp(solver, commitmentMode, "qan-oblig-a"),
  d_cols()
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

// UPLOAD CALLS

// ------------------------------------------------------------
//  MEMBER FUNCTION : uploadObligation
// ------------------------------------------------------------
//  Description  : fill the solver using intermediate calls
//  Role         : host usage
//  Techniques   : (nothing special)
//  Status       : incomplete
// ------------------------------------------------------------

QanObligToSupply_A::index_type
QanObligToSupply_A::uploadObligation
(const double supplyObligation)
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PRELIMINARY CHECKS

  // supply obligations only apply to 'transolve' calls and not
  // to 'capset' calls -- the call type could perhaps be passed
  // thru to here to allow improved scrutiny but at present this
  // information is not forwarded

  double supply = supplyObligation;
  std::string append;
  if ( xeona::isNan(supplyObligation) )
    {
      s_logger->repx(logga::info, "nil obligation received", supplyObligation);
      supply = 0.0;
      append = "-NIL";
    }
  else if ( supplyObligation < 0.0 )
    {
      s_logger->repx(logga::warn, "negative obligation received", supplyObligation);
```

```
      append = "-FAULTY";
    }

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create an exposed variable -- the zero specific costs is, of
  // course, fine
  const int oblCol = pushObj(zeroSpecCosts, lab.str("obligation"));

  // load the local col index into a 1-tuple
  d_cols = boost::make_tuple(oblCol);

  // OBLIGATION BALANCE

  // create the obligation balance
  const int oblRow = pushRhs(supply, svif::E, lab.str("supply-obl" + append));

  // add the exposed variable
  d_cofCount       = pushCof(oblRow, oblCol, +1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'QanObligToSupply_A::uploadObligation'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optjunc.cc
//  file-create-date : Mon 26-Oct-2009 10:26 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : operations OSPs for junctions / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optjunc.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "optjunc.h"          // companion header for this file (place first)

#include "../c/label.h"       // helper class to format solver labels
#include "../c/costs.h"       // cost sets and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : JncSplit2
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : JncSplit2
// ----------------------------------------------------------

JncSplit2::JncSplit2
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  JunctionOsp(solver, commitmentMode, "jnc-2-cable"),
  d_cols()                                  // tuple ctor calls default ctors
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~JncSplit2
// ----------------------------------------------------------

JncSplit2::~JncSplit2()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

JncSplit2::index_type
JncSplit2::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int sockCol  = pushObj(zeroSpecCosts, lab.str("socket"));
  const int cab1Col  = pushObj(zeroSpecCosts, lab.str("cable-1"));
  const int cab2Col  = pushObj(zeroSpecCosts, lab.str("cable-2"));

  // load the local col indexes into a 3-tuple
  d_cols = boost::make_tuple(sockCol,
                             cab1Col,
                             cab2Col);

  // JUNCTION BALANCE

  // create the junction balance
  const int juncRow  = pushRhs(0.0, svif::E, lab.str("junction-bal"));

  // add the exposed variables
  d_cofCount         = pushCof(juncRow, sockCol, +1.0);
  d_cofCount         = pushCof(juncRow, cab1Col, -1.0);
  d_cofCount         = pushCof(juncRow, cab2Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------

JncSplit2::results_type
JncSplit2::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()));
}

// ----------------------------------------------------------
//  CLASS           : JncSplit3
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : JncSplit3
// ----------------------------------------------------------

JncSplit3::JncSplit3
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  JunctionOsp(solver, commitmentMode, "jnc-3-cable"),
  d_cols()                                  // tuple ctor calls default ctors
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~JncSplit3
// ----------------------------------------------------------
```

```
JncSplit3::~JncSplit3()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

JncSplit3::index_type
JncSplit3::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int sockCol  = pushObj(zeroSpecCosts, lab.str("socket"));
  const int cab1Col  = pushObj(zeroSpecCosts, lab.str("cable-1"));
  const int cab2Col  = pushObj(zeroSpecCosts, lab.str("cable-2"));
  const int cab3Col  = pushObj(zeroSpecCosts, lab.str("cable-3"));

  // load the local col indexes into a 4-tuple
  d_cols = boost::make_tuple(sockCol,
                             cab1Col,
                             cab2Col,
                             cab3Col);

  // JUNCTION BALANCE

  // create the junction balance
  const int juncRow  = pushRhs(0.0, svif::E, lab.str("junction-bal"));

  // add the exposed variables
  d_cofCount         = pushCof(juncRow, sockCol, +1.0);
  d_cofCount         = pushCof(juncRow, cab1Col, -1.0);
  d_cofCount         = pushCof(juncRow, cab2Col, -1.0);
  d_cofCount         = pushCof(juncRow, cab3Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------

JncSplit3::results_type
JncSplit3::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()));
}

// ----------------------------------------------------------
//  CLASS           : JncJoin2
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : JncJoin2
// ----------------------------------------------------------

JncJoin2::JncJoin2
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
```

```
  JunctionOsp(solver, commitmentMode, "jnc-2-socket"),
  d_cols()                                    // tuple ctor calls default ctors
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~JncJoin2
// ----------------------------------------------------------

JncJoin2::~JncJoin2()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

JncJoin2::index_type
JncJoin2::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int cableCol  = pushObj(zeroSpecCosts, lab.str("cable"));
  const int sock1Col  = pushObj(zeroSpecCosts, lab.str("socket-1"));
  const int sock2Col  = pushObj(zeroSpecCosts, lab.str("socket-2"));

  // load the local col indexes into a 3-tuple
  d_cols = boost::make_tuple(cableCol,
                             sock1Col,
                             sock2Col);

  // JUNCTION BALANCE

  // create the junction balance
  const int juncRow  = pushRhs(0.0, svif::E, lab.str("junction-bal"));

  // add the exposed variables
  d_cofCount         = pushCof(juncRow, cableCol, -1.0);
  d_cofCount         = pushCof(juncRow, sock1Col, +1.0);
  d_cofCount         = pushCof(juncRow, sock2Col, +1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------

JncJoin2::results_type
JncJoin2::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()));
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optnode.cc
//  file-create-date : Tue 03-Nov-2009 19:49 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : node optimization sub-problems for LMP nodes / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optnode.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "optnode.h"          // companion header for this file (place first)

#include "../c/label.h"       // helper class to format solver labels
#include "../c/costs.h"       // cost sets and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : LmpCab1
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : LmpCab1
// ----------------------------------------------------------

LmpCab1::LmpCab1
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  LmpNodeOsp(solver, commitmentMode, "lmp-cable+bilateral"),
  d_cols(),                                // tuple ctor calls default ctors
  d_row(0)
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~LmpCab1
// ----------------------------------------------------------

LmpCab1::~LmpCab1()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}
```

```
// -------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// -------------------------------------------------------

LmpCab1::index_type
LmpCab1::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int cab1Col = pushObj(zeroSpecCosts, lab.str("cable"));
  const int soc1Col = pushObj(zeroSpecCosts, lab.str("socket-bi"));

  // load the local col indexes into a 2-tuple
  d_cols = boost::make_tuple(cab1Col,
                             soc1Col);

  // NODE BALANCE

  // bidirectionalize the socket
  const int bndChanges = openBnds(soc1Col);

  // defensive programming (for this and 'LmpSoc1')
  switch ( bndChanges )
    {
    case 0:
      s_logger->repx(logga::warn, "bidirectionalize failed, changes", bndChanges);
      break;
    case 1:                                  // hopefully 0.0 to -inf
      // do nothing okay
      break;
    case 2:
      s_logger->repx(logga::warn, "bidirectionalize failed, changes", bndChanges);
      break;
    default:
      std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
      break;
    }

  // create the node balance
  const int nodeRow = pushRhs(0.0, svif::E, lab.str("node-bal"));

  // load the local row index into an int
  d_row = nodeRow;

  // add the exposed variables
  d_cofCount        = pushCof(nodeRow, cab1Col, +1.0);
  d_cofCount        = pushCof(nodeRow, soc1Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// -------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// -------------------------------------------------------

LmpCab1::results_type
LmpCab1::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()),
                           downloadSlack(d_row));
```

```
}

// ----------------------------------------------------------
//  CLASS           : LmpSoc1
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : LmpSoc1
// ----------------------------------------------------------

LmpSoc1::LmpSoc1
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  LmpNodeOsp(solver, commitmentMode, "lmp-socket+bilateral"),
  d_cols(),                                 // tuple ctor calls default ctors
  d_row(0)
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~LmpSoc1
// ----------------------------------------------------------

LmpSoc1::~LmpSoc1()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

LmpSoc1::index_type
LmpSoc1::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int cab1Col = pushObj(zeroSpecCosts, lab.str("cable-bi"));
  const int soc1Col = pushObj(zeroSpecCosts, lab.str("socket"));

  // load the local col indexes into a 2-tuple
  d_cols = boost::make_tuple(cab1Col,
                             soc1Col);

  // NODE BALANCE

  // bidirectionalize the socket
  const int bndChanges = openBnds(cab1Col);

  // defensive programming
  switch ( bndChanges )
    {
    case 0:
      s_logger->repx(logga::warn, "bidirectionalize failed, changes", bndChanges);
      break;
    case 1:                                 // hopefully 0.0 to -inf
      // do nothing okay
      break;
    case 2:
      s_logger->repx(logga::warn, "bidirectionalize failed, changes", bndChanges);
      break;
    default:
      std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
      break;
    }

  // create the node balance
```

```
  const int nodeRow = pushRhs(0.0, svif::E, lab.str("node-bal"));

  // load the local row index into an int
  d_row = nodeRow;

  // add the exposed variables
  d_cofCount        = pushCof(nodeRow, cab1Col, +1.0);
  d_cofCount        = pushCof(nodeRow, soc1Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------

LmpSoc1::results_type
LmpSoc1::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()),
                           downloadSlack(d_row));
}

// ----------------------------------------------------------
//  CLASS           : LmpNul2
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : LmpNul2
// ----------------------------------------------------------

LmpNul2::LmpNul2
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  LmpNodeOsp(solver, commitmentMode, "lmp-two-bilateral"),
  d_cols(),                                  // tuple ctor calls default ctors
  d_row(0)
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~LmpNul2
// ----------------------------------------------------------

LmpNul2::~LmpNul2()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

LmpNul2::index_type
LmpNul2::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int cab1Col = pushObj(zeroSpecCosts, lab.str("cable-bi"));
  const int soc1Col = pushObj(zeroSpecCosts, lab.str("socket-bi"));
```

```
  // load the local col indexes into a 2-tuple
  d_cols = boost::make_tuple(cab1Col,
                             soc1Col);

  // NODE BALANCE

  // bidirectionalize the cable and socket
  openBnds(cab1Col);
  openBnds(soc1Col);

  // create the node balance
  const int nodeRow = pushRhs(0.0, svif::E, lab.str("node-bal"));

  // load the local row index into an int
  d_row = nodeRow;

  // add the exposed variables
  d_cofCount        = pushCof(nodeRow, cab1Col, +1.0);
  d_cofCount        = pushCof(nodeRow, soc1Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------

LmpNul2::results_type
LmpNul2::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()),    // either would do
                           downloadSlack(d_row));
}

// ----------------------------------------------------------
//  CLASS           : LmpCab2
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : LmpCab2
// ----------------------------------------------------------

LmpCab2::LmpCab2
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  LmpNodeOsp(solver, commitmentMode, "lmp-cable+two-bilateral"),
  d_cols(),                                 // tuple ctor calls default ctors
  d_row(0)
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~LmpCab2
// ----------------------------------------------------------

LmpCab2::~LmpCab2()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

LmpCab2::index_type
LmpCab2::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE
```

```
  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int cab1Col = pushObj(zeroSpecCosts, lab.str("cable-1"));
  const int cab2Col = pushObj(zeroSpecCosts, lab.str("cable-2-bi"));
  const int soc1Col = pushObj(zeroSpecCosts, lab.str("socket-bi"));

  // load the local col indexes into a 2-tuple
  d_cols = boost::make_tuple(cab1Col,
                             cab2Col,
                             soc1Col);

  // NODE BALANCE

  // bidirectionalize the cable and socket
  openBnds(cab1Col);
  openBnds(cab2Col);
  openBnds(soc1Col);

  // create the node balance
  const int nodeRow = pushRhs(0.0, svif::E, lab.str("node-bal"));

  // load the local row index into an int
  d_row = nodeRow;

  // add the exposed variables
  d_cofCount        = pushCof(nodeRow, cab1Col, +1.0);
  d_cofCount        = pushCof(nodeRow, cab2Col, +1.0);
  d_cofCount        = pushCof(nodeRow, soc1Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ------------------------------------------------------------

LmpCab2::results_type
LmpCab2::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<0>()),
                           downloadSlack(d_row));
}

// ------------------------------------------------------------
//  CLASS           : LmpSoc2
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : LmpSoc2
// ------------------------------------------------------------

LmpSoc2::LmpSoc2
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  LmpNodeOsp(solver, commitmentMode, "lmp-socket+two-bilateral"),
  d_cols(),                                  // tuple ctor calls default ctors
  d_row(0)
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~LmpSoc2
// ------------------------------------------------------------

LmpSoc2::~LmpSoc2()
```

```
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------

//    <int,                                     // cable-bi    (bidirectional)
//     int,                                     // socket-1    (normal)
//     int>                                     // socket-2-bi (single, bidirectional)

LmpSoc2::index_type
LmpSoc2::uploadEngineering()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // EXPOSED VARIABLES

  // create some variables covering input and output -- the zero
  // specific costs are, of course, fine
  const int cab1Col = pushObj(zeroSpecCosts, lab.str("cable-bi"));
  const int soc1Col = pushObj(zeroSpecCosts, lab.str("socket-1"));
  const int soc2Col = pushObj(zeroSpecCosts, lab.str("socket-2-bi"));

  // load the local col indexes into a 2-tuple
  d_cols = boost::make_tuple(cab1Col,
                             soc1Col,
                             soc2Col);

  // NODE BALANCE

  // bidirectionalize the cable and socket
  openBnds(cab1Col);
  openBnds(soc2Col);

  // create the node balance
  const int nodeRow = pushRhs(0.0, svif::E, lab.str("node-bal"));

  // load the local row index into an int
  d_row = nodeRow;

  // add the exposed variables
  d_cofCount        = pushCof(nodeRow, cab1Col, +1.0);
  d_cofCount        = pushCof(nodeRow, soc1Col, -1.0);
  d_cofCount        = pushCof(nodeRow, soc2Col, -1.0);

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------

LmpSoc2::results_type
LmpSoc2::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // active code
  return boost::make_tuple(downloadVar(d_cols.get<1>()),
                           downloadSlack(d_row));
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optops.cc
//  file-create-date : Fri 17-Oct-2008 14:41 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : operations OSPs for technical assets / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optops.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "optops.h"           // companion header for this file (place first)

#include "../c/label.h"       // helper class to format solver labels
#include "../c/costs.h"       // cost sets and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <cmath>              // C-style maths, ceil(), floor(), sqrt()

#include <boost/format.hpp>   // printf style formatting
#include <boost/ref.hpp>      // pass references to STL algorithms and such

//  CODE

// ----------------------------------------------------------
//  notes           : Operations optimization sub-problems
// ----------------------------------------------------------
//
//  Design notes generally applicable to operations OSPs
//
//      Use of input and output balances
//
//          Setting up explicit balances to handle input and
//          output is clearly redundant for a one blockette OSP.
//
//          Notwithstanding, this practice should be honored
//          because the resulting code will be easier to read,
//          maintain, and extend if written in a consistent way.
//
//      Return capture variables
//
//          The following base class 'OptimSubProb' variables may
//          be used to capture push returns (and similar) in
//          statements where no subsequent usage is envisaged:
//
//              d_colTrack
//              d_rowTrack
```

```
//              d_cofCount
//
//      Separation of engineering and specific costs uploads
//
//          For most OSPs, the upload of specific costs and of
//          the engineering characterizations are separated and
//          should be processed in this order:
//
//              uploadEngineering   - engineering characterization
//              uploadShortrunCosts - specific cost data (optional)
//
//          As indicated, 'OperationsOsp::uploadShortrunCost' is
//          normally relied upon for loading specific costs, but
//          its use is optional -- in which case the
//          'OperationsOsp::uploadEngineering' default (zero by
//          convention) will be employed.
//
//          More complex designs are possible but they would
//          require both aspects to be treated in the same
//          routine.  In addition, the user-supplied model data
//          overhead would need to increase as a result.
//
//      Duty
//
//          The definition of duty is left to the host entity
//          (and perhaps ultimately the modeler), as is the
//          explicit coupling of duty between the technical asset
//          and the asset operator.  This is an important
//          concept!
//
//          For a transformation block entity with just one
//          output stream, duty invariably equates to that
//          output.  In other words, the concept of elected duty
//          only becomes a model choice issue for multi-product
//          assets (such as combined heat and power).
//
//      Boost tuples and 'boost::make_tuple' usage
//
//          By default, 'boost::make_tuple' holds non-const,
//          non-reference values -- even if the argument in
//          question is a reference or a const reference.
//
//          In this file, we may want to hold const references in
//          tuples -- which requires the use of 'boost::cref'
//          from header <boost/ref.hpp>.
//
//          See Karlsson (2006 pp216-217) for more details on
//          'cref.'  Becker (2007 pp3-22) also covers tuples.
//
//      Standard labels for use in 'uploadEngineering' definitions
//
//          OSP authors should deploy the following labels where
//          appropriate:
//
//          // initial reporting
//          // PREAMBLE
//          // INTEGRITY CHECKS
//          // EXPOSED VARIABLES
//          // FACTOR AND OUTPUT BALANCES
//          // BLOCKETTES, etc
//          // GLOBALIZE AND RETURN
//
// ------------------------------------------------------------

// ------------------------------------------------------------
//  CLASS           : OpsTransmisson_A (HV transmission)
// ------------------------------------------------------------

// CREATORS

// ------------------------------------------------------------
//  MEMBER FUNCTION : OpsTransmisson_A
// ------------------------------------------------------------

OpsTransmission_A::OpsTransmission_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  OperationsOsp(solver, commitmentMode, "ops-transmission-a"),
  d_cols(),                                 // tuple ctor calls default ctors
  d_injectCapacity(0.0)
{
```

```
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~OpsTransmisson_A
// ----------------------------------------------------------

OpsTransmission_A::~OpsTransmission_A()
{
  s_logger->repx(logga::adhc, "destructor call, description", d_ospDesc);
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering (pre-processing)
// ----------------------------------------------------------
//  Description  : problem building, higher-level wrapper to principal call
//  Role         : host call, convenience
//  Techniques   : physics
//  Status       : complete (but see also principal call status)
//
//  DC power flow model used
//
//                R       I
//      +   o---VVVVVV------o
//      |
//    V |
//      |       V                 electromotive force [V]
//      =       I                 current [A]
//              R                 resistance [ohm]
//              P      = V.I      power carried in conductor
//              Ploss  = R.I^2    power loss in resistor
//              deltaV = R.I      voltage drop across resistor
//
//  the relative losses Ploss/P at capacity are used here
//
//                R
//     Ploss/P = ---      and at capacity P = V.Icap
//                V^2
//
//  In passing, note also that a fixed frequency AC power flow
//  model has been developed for LMP pricing, but this is much
//  more complicated.  For instance, see work by University of
//  New South Wales, Australia.
//
//  Typical specific resistance values
//
//      voltage      ohms/m
//      -------------------
//      500kV        30e-06
//      230kV       100e-06
//      115kV       170e-06
//
//  Source: Phillips, Drew.  2004.  Nodal pricing basics -- OHPs.
//  IMO independent electricity market operator.
//  LMP_NodalBasics_2004jan14.pdf [file downloaded from web in
//  Feb-2008].  Reverse engineered from slide 12.  See also my
//  OpenOffice spreadsheet (*line-losses*.ods).
//
// ----------------------------------------------------------

OpsTransmission_A::index_type                // global cols
OpsTransmission_A::uploadEngineering
(const double injectCapacity,                // in unprefixed SI units [W]
 const double voltage,                       // in unprefixed SI units [V]
 const double ohmsPerMetre,                  // in unprefixed SI units [ohm/m]
 const double length,                        // in unprefixed SI units [m]
 const int    discretization)                // number of discretization steps
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // process arguments
  const double ohms         = ohmsPerMetre * length;        // DC resistance
  const double slope        = ohms / std::pow(voltage, 2);  // R/V^2, refer <cmath>
  const double maxRelLosses = slope * injectCapacity;       // y upper (y lower is zero)
  if ( maxRelLosses >= 1.0 )
    {
      // could also check to see if voltage drop exceeds input voltage
      s_logger->repx(logga::warn, "aphysical line characterization", "");
```

```
      std::ostringstream put;
      put << "  line characterization aphysical : " << d_label  << "\n"
          << "    inject capacity  : " << injectCapacity        << "\n"
          << "    voltage          : " << voltage               << "\n"
          << "    ohms per metre   : " << ohmsPerMetre          << "\n"
          << "    length           : " << length                << "\n"
          << "    max rel losses   : " << maxRelLosses          << "\n";
      s_logger->putx(logga::dbug, put);
    }

  // call principal function
  return uploadEngineering(injectCapacity,
                           maxRelLosses,
                           discretization);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering (principal call)
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : step-wise discretization
//  Status       : first-pass complete but needs careful checking
// ----------------------------------------------------------

OpsTransmission_A::index_type                // global cols
OpsTransmission_A::uploadEngineering
(const double injectCapacity,                // input capacity
 const double maxRelLosses,                  // relative losses at capacity
 const int    discretization)                // number of discretization steps
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // set persistant variables (from this class 'OpsTransmission_A')
  d_injectCapacity = injectCapacity;

  // process arguments (note the staircase analogy)
  const double tread = injectCapacity / discretization;     // x-axis discretization
  const double riser = maxRelLosses   / discretization;     // y-axis discretization

  // EXPOSED VARIABLES

  // create two variables covering input and output -- the zero
  // specific costs can later be rectified by
  // 'uploadShortrunCosts' but, in all likelihood, will simply
  // remain zero
  const int cabCol = pushObj(zeroSpecCosts, lab.str("cable-bi"));
  const int socCol = pushObj(zeroSpecCosts, lab.str("socket-bi"));

  // load the local indexes into a 2-tuple
  d_cols = boost::make_tuple(cabCol,
                             socCol);

  // CABLE AND SOCKET BALANCES

  // bidirectionalize the cable and socket
  int bndChanges = -1;                       // nonsensical value
  bndChanges     = openBnds(cabCol);
  bndChanges     = openBnds(socCol);

  // create cable and socket balances
  const int cabRow = pushRhs(0.0, svif::E, lab.str("cable-bal"));
  const int socRow = pushRhs(0.0, svif::E, lab.str("socket-bal"));

  // add the exposed variables
  d_cofCount = pushCof(cabRow, cabCol, +1.0);
  d_cofCount = pushCof(socRow, socCol, -1.0);

  // FORWARD BLOCKETTES

  double loss   = 0;                         // normalized loss, y-value
  double lower  = 0;                         // current lower capacity, x-value
```

```
  double upper  = 0;                          // current upper capacity, x-value

  for ( int i = 1;                            // loop the blockettes
        i    <= discretization;
        ++i )
    {
      loss  = (i - 0.5) * riser;              // forms 1/2, 3/2, 5/2, ..
      lower = 0.0;
      upper = tread;

      lab << boost::format("%s%02d") % "fwd" % i;      // "fwd00"

      // two blockette variables
      const int  inCol = pushObj(0.0, lab.str("in"));
      const int outCol = pushObj(0.0, lab.str("out"));

      // connect to the input and output balances
      d_cofCount = pushCof(cabRow,  inCol, -1.0);
      d_cofCount = pushCof(socRow, outCol, +1.0);

      // define the lower and upper constraints on input
      const int lowerRow = pushRhs(lower, svif::G, lab.str("lower"));
      const int upperRow = pushRhs(upper, svif::L, lab.str("upper"));
      d_cofCount         = pushCof(lowerRow, inCol, +1.0);
      d_cofCount         = pushCof(upperRow, inCol, +1.0);

      // define the input/output equality
      const int iorRow = pushRhs(0.0, svif::E, lab.str("ior"));
      d_cofCount       = pushCof(iorRow,  inCol, 1.0 - loss);
      d_cofCount       = pushCof(iorRow, outCol, -1.0);

      lab.trim(1);                            // remove the last labelette
    }

  // BACKWARD BLOCKETTES

  loss   = 0;                                 // reset
  lower  = 0;                                 // reset
  upper  = 0;                                 // reset

  for ( int i = 1;                            // loop the blockettes
        i    <= discretization;
        ++i )
    {
      loss  = (i - 0.5) * riser;              // forms 1/2, 3/2, 5/2, ..
      lower = 0.0;
      upper = tread;

      lab << boost::format("%s%02d") % "bak" % i;      // "bak00"

      // two blockette variables
      const int inCol  = pushObj(0.0, lab.str("in"));
      const int outCol = pushObj(0.0, lab.str("out"));

      // connect to the input and output balances
      d_cofCount = pushCof(cabRow, outCol, +1.0);
      d_cofCount = pushCof(socRow,  inCol, -1.0);

      // define the lower and upper constraints on input
      const int lowerRow = pushRhs(lower, svif::G, lab.str("lower"));
      const int upperRow = pushRhs(upper, svif::L, lab.str("upper"));
      d_cofCount         = pushCof(lowerRow, inCol, +1.0);
      d_cofCount         = pushCof(upperRow, inCol, +1.0);

      // define the input/output equality
      const int iorRow = pushRhs(0.0, svif::E, lab.str("ior"));
      d_cofCount       = pushCof(iorRow,  inCol, 1.0 - loss);
      d_cofCount       = pushCof(iorRow, outCol, -1.0);

      lab.trim(1);                            // remove the last labelette
    }

  // additional reporting as appropriate
  // YEEK 24 CODE (set by '--yeek')
  if ( xeona::yeek == 24 || xeona::yeek == 1 )
    {
      const std::string func = __func__;     // a preprocessor macro
      reportBuildIntegrity(func + " (principal call)");
    }

  // GLOBALIZE AND RETURN
```

```
  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'OpsTransmission_A::uploadEngineering'

// DOWNLOAD SOLUTION CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ----------------------------------------------------------
//  Description  : solution recovery
//  Role         : host call
//  Techniques   : directionality calculations
//  Status       : complete
// ----------------------------------------------------------

OpsTransmission_A::results_type
OpsTransmission_A::downloadSolution() const
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // identify transmission results
  const double cab = downloadVar(d_cols.get<0>());     // bi-directional
  const double soc = downloadVar(d_cols.get<1>());     // bi-directional

  // integrity check
  if ( (cab * soc) < 0.0 )                  // one and only one is negative
    {
      s_logger->repx(logga::warn, "integrity check failure", "unmatched signs");
      std::ostringstream put;
      put << "  cable flow  : " << std::showpos << cab << "\n"
          << "  socket flow : " << std::showpos << soc << "\n";
      s_logger->putx(logga::xtra, put);
    }

  // process secondary quantities
  const bool direction = cab < 0 ? false : true;
  double inject = -1.0;                     // current injection, always positive
  double exit   = -1.0;                     // current exit, always positive
  if ( direction )                          // positive commodity flow encountered
    {
      inject = +cab;
      exit   = +soc;
    }
  else                                      // negative commodity flow encountered
    {
      inject = -soc;
      exit   = -cab;
    }

  double relativeDuty = inject / d_injectCapacity;
  if ( relativeDuty == getNaN() ) relativeDuty = 0.0;  // protect against div-by-zero
  double relativeLoss = (inject - exit) / inject;
  if ( relativeLoss == getNaN() ) relativeLoss = 0.0;  // protect against div-by-zero

  // additional reporting as appropiate
  // YEEK 11 CODE (set by '--yeek')
  if ( xeona::yeek == 11 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  transmission OSP values:"                                  << "\n"
          << "    cable         : " << cab                                 << "\n"
          << "    socket        : " << soc                                 << "\n"
          << "    inject flow   : " << inject                              << "\n"
          << "    exit flow     : " << exit                                << "\n"
          << "    relative duty : " << relativeDuty                        << "\n"
          << "    relative loss : " << relativeLoss                        << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // return
  return boost::make_tuple(direction,
                           cab,
                           soc,
                           relativeDuty,
                           relativeLoss);

} // function 'downloadSolution'
```

```
// ----------------------------------------------------------
//  CLASS           : OpsFac1Out1_A (shutdown mode)
// ----------------------------------------------------------

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : OpsFac1Out1_A
// ----------------------------------------------------------

OpsFac1Out1_A::OpsFac1Out1_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  OperationsOsp(solver, commitmentMode, "ops-f1o1-shutdown-a"),
  d_cols()                                  // tuple ctor calls default ctors
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~OpsFac1Out1_A
// ----------------------------------------------------------

OpsFac1Out1_A::~OpsFac1Out1_A()
{
  s_logger->repx(logga::adhc, "destructor call, description", d_ospDesc);
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : binary variable for shutdown mode
//  Status       : first-pass complete but needs careful checking
//
//  Design notes
//
//      Shutdown mode operation.
//
//          This OSP supports shutdown mode operation, as tripped
//          by the 'prodLoBound' value.  It also employs constant
//          marginal efficiency.  See my (Robbie Morrison) PhD
//          thesis for the underlying equations.
//
//      kg-to-J wrapper
//
//          Note this wrapper which also requires mass-specific
//          enthalpy data.
//
//      Abnormal conditions
//
//          As an extension, one could relax the trip condition
//          for times when the original "tripped" problem is
//          infeasible.
//
// ----------------------------------------------------------

// kg-to-J wrapper

OpsFac1Out1_A::index_type                   // global cols
OpsFac1Out1_A::uploadEngineering            // kg-to-J form
(const double prodLoBound,                  // lower output below which the asset trips
 const double prodHiBound,                  // upper output
 const double marginalEfficiency,           // slope (as decimal not percentage)
 const double fuelNoload,                   // fuel usage on idle (input-axis intercept)
 const double fuelAncillary,                // fuel usage on shutdown (for ancillaries)
 const double rampLoSize,                   // ramp down restriction
 const double rampHiSize,                   // ramp up restriction
 const double specEnthalpy)                 // kg->J conversion
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "kg-to-J form");

  // INTEGRITY CHECKS

  // very basic range checking -- with both zero and unity
  // efficiency being valid (also caters for most cases when a
```

```
  // percentage is mistakenly used)
  if ( marginalEfficiency < 0.0 || marginalEfficiency > 1.0 )
    {
      s_logger->repx(logga::rankJumpy,
                     "aphysical marginal efficiency",
                     marginalEfficiency);
    }
  // specific enthalpy typically 20e06 (delete or modify code if no longer appropriate)
  if ( specEnthalpy < 1.0e06 || specEnthalpy > 100.0e06)
    {
      s_logger->repx(logga::rankJumpy,
                     "surprising specific enthalpy value",
                     specEnthalpy);
    }

  // PRINCIPAL CALL

  // under the current data specification, the last two arguments
  // do not need to be be multiplied by 'specEnthalpy'
  return uploadEngineering(prodLoBound,
                           prodHiBound,
                           marginalEfficiency * specEnthalpy,
                           fuelNoload         / specEnthalpy,
                           fuelAncillary      / specEnthalpy,
                           rampLoSize,
                           rampHiSize);

} // function 'uploadEngineering'

// J-to-J form

OpsFac1Out1_A::index_type                    // global cols
OpsFac1Out1_A::uploadEngineering             // J-to-J form
(const double prodLoBound,                   // lower output below which the asset trips
 const double prodHiBound,                   // upper output
 const double marginalEfficiency,            // slope (as decimal not percentage)
 const double fuelNoload,                    // fuel usage on idle (input-axis intercept)
 const double fuelAncillary,                 // fuel usage on shutdown (for ancillaries)
 const double rampLoSize,                    // ramp down restriction
 const double rampHiSize)                    // ramp up restriction
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "J-to-J form");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // INTEGRITY CHECKS

  // very basic range checking, noting that the marginal
  // efficiency may be multiplied by the specific enthalpy prior
  // to this call
  if ( marginalEfficiency < 0.0 )
    {
      s_logger->repx(logga::rankJumpy,
                     "aphysical marginal efficiency",
                     marginalEfficiency);
    }
  // advisory message
  if ( fuelAncillary > fuelNoload )
    {
      std::ostringstream oss;
      oss << fuelAncillary << " : " << fuelNoload;
      s_logger->repx(logga::rankJumpy,
                     "advise fuel ancillary > fuel noload",
                     oss.str());
    }

  // EXPOSED VARIABLES

  // create two variables covering input and output -- the zero
  // specific costs can later be rectified by 'uploadShortrunCosts'
  const int fac1Col  = pushObj(zeroSpecCosts, lab.str("factor"));
  const int out1Col  = pushObj(zeroSpecCosts, lab.str("output"));

  // create a binary variable covering shutdown mode, whereby
```

```
  // value 0 (implicit 'false') = shutdown
  const int tripCol  = pushObj(zeroSpecCosts, lab.str("trip-if-zero"));
  d_colTrack         = markBinary(tripCol);

  // load the local col indexes into a 3-tuple
  d_cols = boost::make_tuple(fac1Col,
                             out1Col,
                             tripCol);

  // FACTOR AND OUTPUT BALANCES

  // create input and output balances
  const int fac1Row   = pushRhs(0.0, svif::E, lab.str("factor-bal"));
  const int out1Row   = pushRhs(0.0, svif::E, lab.str("output-bal"));

  // add the exposed variables
  d_cofCount          = pushCof(fac1Row, fac1Col, +1.0);
  d_cofCount          = pushCof(out1Row, out1Col, -1.0);

  // BLOCKETTE IO
  // note that blockette Io is the sole blockette in this OSP

  // create in and out variables for blockette Io
  const int  inIoCol  = pushObj(zeroSpecCosts, lab.str("block-io-in"));
  const int outIoCol  = pushObj(zeroSpecCosts, lab.str("block-io-out"));

  // connect to the input and output balances
  d_cofCount          = pushCof(fac1Row,  inIoCol, -1.0);
  d_cofCount          = pushCof(out1Row, outIoCol, +1.0);

  // define the input/output equality for blockette Io
  const int effRow    = pushRhs(fuelAncillary, svif::E, lab.str("efficiency"));
  d_cofCount          = pushCof(effRow,  inIoCol, +1.0);
  d_cofCount          = pushCof(effRow, outIoCol, -1.0 / marginalEfficiency);
  d_cofCount          = pushCof(effRow,  tripCol, fuelAncillary - fuelNoload);

  // define the lo and hi bounds for blockette Io -- note that
  // these calls are  complicated by the shutdown mode logic
  const int loRow     = pushRhs(0.0, svif::G, lab.str("lo-bound"));
  d_cofCount          = pushCof(loRow,  tripCol, -prodLoBound);
  d_cofCount          = pushCof(loRow, outIoCol, +1.0);

  const int hiRow     = pushRhs(0.0, svif::L, lab.str("hi-bound"));
  d_cofCount          = pushCof(hiRow,  tripCol, -prodHiBound);
  d_cofCount          = pushCof(hiRow, outIoCol, +1.0);

  // add the ramp rate restrictions
  const int rampLoRow = pushRhs(rampLoSize, svif::G, lab.str("lo-ramp-restraint"));
  const int rampHiRow = pushRhs(rampHiSize, svif::L, lab.str("hi-ramp-restraint"));
  d_cofCount          = pushCof(rampLoRow, outIoCol, +1.0);
  d_cofCount          = pushCof(rampHiRow, outIoCol, +1.0);

  // YEEK 24 CODE (set by '--yeek')
  if ( xeona::yeek == 24 || xeona::yeek == 1 )
    {
      const std::string func = __func__;     // a preprocessor macro
      reportBuildIntegrity(func + " (J-to-J call)");
    }

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'uploadEngineering'

// DOWNLOAD SOLUTION CALLS

// ------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ------------------------------------------------------
//  Description  : solution recovery
//  Role         : host call
//  Techniques   : 'boost::make_tuple'
//  Status       : complete
// ------------------------------------------------------

OpsFac1Out1_A::results_type
OpsFac1Out1_A::downloadSolution() const
{
  // YEEK 3 CODE (set by '--yeek')
```

```
  if ( xeona::yeek == 3 || xeona::yeek == 1 )
    {
      s_logger->repx(logga::yeek, "entering member function", "");
      const int    gol = d_cols.get<2>();
      const double var = downloadVar(gol);
      const bool   bol = static_cast<bool>(var);
      std::ostringstream put;
      put << "  gol    : " << gol << "\n"
          << "  double : " << var << "\n"
          << "  bool   : " << bol << "\n"
          << std::boolalpha
          << "  bool   : " << bol << "\n";
      s_logger->putx(logga::yeek, put);
    }

  return boost::make_tuple(downloadVar(d_cols.get<0>()),
                           downloadVar(d_cols.get<1>()),
                           static_cast<bool>(downloadVar(d_cols.get<2>()))); // [1]

  // [1] a 'double' { 0.0, 1.0 } to 'bool' { 'false', 'true' } conversion

} // function 'downloadSolution'

// ----------------------------------------------------------
//  CLASS           : OpsFac0Out1_A (source)
// ----------------------------------------------------------

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : OpsFac0Out1_A
// ----------------------------------------------------------

OpsFac0Out1_A::OpsFac0Out1_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  OperationsOsp(solver, commitmentMode, "ops-f0o1-a"),
  d_cols()                                  // tuple ctor calls default ctors
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~OpsFac0Out1_A
// ----------------------------------------------------------

OpsFac0Out1_A::~OpsFac0Out1_A()
{
  s_logger->repx(logga::adhc, "destructor call, describtion", d_ospDesc);
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : first-pass complete but needs careful checking
// ----------------------------------------------------------

OpsFac0Out1_A::index_type
OpsFac0Out1_A::uploadEngineering
(const double prodLoBound,                  // lower output (typically zero)
 const double prodHiBound)                  // upper output
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);                       // entity identifier

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // INTEGRITY CHECKS

  // very basic integrity checking
```

```
  if ( prodLoBound > prodHiBound ||
       prodLoBound < 0.0 )
    {
      std::ostringstream oss;
      oss << prodLoBound << " : " << prodHiBound;
      s_logger->repx(logga::warn, "aphysical factor bounds, lo : hi", oss.str());
    }

  // EXPOSED VARIABLES

  // create one variable covering output (and thus the only
  // choice on offer for duty) -- the zero specific costs can
  // later be rectified by 'uploadShortrunCosts'
  const int prodCol = pushObj(zeroSpecCosts, lab.str("output"));

  // load the local index into a 1-tuple
  d_cols            = boost::make_tuple(prodCol);

  // OUTPUT BALANCE

  // create an output balance
  const int outRow  = pushRhs(0.0, svif::E, lab.str("output-bal"));

  // add the exposed variable
  d_cofCount        = pushCof(outRow, prodCol, -1.0);

  // SOURCE BLOCKETTE

  // note that the source blockette is the sole blockette in this
  // OSP -- now create an out variable for the source blockette
  const int outCol  = pushObj(zeroSpecCosts, lab.str("source-out"));

  // connect to the output balance
  d_cofCount        = pushCof(outRow, outCol, +1.0);

  // define the input/output equality for the source blockette
  // (not required)

  // define the lo and hi bounds for the source blockette
  const int loRow   = pushRhs(prodLoBound, svif::G, lab.str("lo-bound"));
  d_cofCount        = pushCof(loRow, outCol, +1.0);

  const int hiRow   = pushRhs(prodHiBound, svif::L, lab.str("hi-bound"));
  d_cofCount        = pushCof(hiRow, outCol, +1.0);

  // additional reporting as appropriate
  // YEEK 24 CODE (set by '--yeek')
  if ( xeona::yeek == 24 || xeona::yeek == 1 )
    {
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      const std::string func = __func__;     // a preprocessor macro
      reportBuildIntegrity(func + " (lo/hi output call)");
    }

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'uploadEngineering'

// DOWNLOAD SOLUTION CALLS

// ------------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ------------------------------------------------------------
//  Description  : solution recovery
//  Role         : host call
//  Techniques   : 'boost::make_tuple'
//  Status       : complete
// ------------------------------------------------------------

OpsFac0Out1_A::results_type
OpsFac0Out1_A::downloadSolution() const
{
  // additional reporting
  const int col      = d_cols.get<0>();
  const double value = downloadVar(col);
  std::ostringstream oss;
  oss << "value for col " << col;
  s_logger->repx(logga::adhc, oss.str(), value);
```

```
  // main call
  return boost::make_tuple(downloadVar(d_cols.get<0>()));
}

// ----------------------------------------------------------
//  CLASS           : OpsFac1Out0_A (sink)
// ----------------------------------------------------------

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : OpsFac1Out0_A
// ----------------------------------------------------------

OpsFac1Out0_A::OpsFac1Out0_A
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode) :
  OperationsOsp(solver, commitmentMode, "ops-f1o0-a"),
  d_cols()                                  // tuple ctor calls default ctors
{
  s_logger->repx(logga::adhc, "constructor call, description", d_ospDesc);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~OpsFac1Out0_A
// ----------------------------------------------------------

OpsFac1Out0_A::~OpsFac1Out0_A()
{
  s_logger->repx(logga::adhc, "destructor call, description", d_ospDesc);
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : first-pass complete but needs careful checking
// ----------------------------------------------------------

OpsFac1Out0_A::index_type
OpsFac1Out0_A::uploadEngineering
(const double takeFixed)                    //  fixed input (can be zero)
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);                       // entity identifier

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // INTEGRITY CHECKS

  // very basic integrity checking
  if ( takeFixed < 0.0 )
    {
      s_logger->repx(logga::warn, "aphysical factor bound, fixed", takeFixed);
    }

  // EXPOSED VARIABLES

  // create one variable covering input (and thus the only choice
  // on offer for duty) -- the zero specific costs can later be
  // rectified by 'uploadShortrunCosts'
  const int fuelCol = pushObj(zeroSpecCosts, lab.str("factor"));

  // load the local index into a 1-tuple
  d_cols            = boost::make_tuple(fuelCol);

  // FACTOR BALANCE

  // create an input balance
  const int inRow   = pushRhs(0.0, svif::E, lab.str("factor-bal"));
```

```
  // add the exposed variable
  d_cofCount        = pushCof(inRow, fuelCol, -1.0);

  // SINK BLOCKETTE

  // note that the sink blockette is the sole blockette in this
  // OSP -- now create an in variable for the sink blockette
  const int sinkCol   = pushObj(zeroSpecCosts, lab.str("sink-in"));

  // connect to the input balance
  d_cofCount        = pushCof(inRow, sinkCol, +1.0);

  // define the input/output equality for the source blockette
  // (not required)

  // define the fixed bound for the source blockette
  const int fixRow  = pushRhs(takeFixed, svif::E, lab.str("fixed-bound"));
  d_cofCount        = pushCof(fixRow, sinkCol, +1.0);

  // YEEK 24 CODE (set by '--yeek')
  if ( xeona::yeek == 24 || xeona::yeek == 1 )
    {
      const std::string func = __func__;     // a preprocessor macro
      reportBuildIntegrity(func + " (fixed take call)");
    }

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'uploadEngineering'

// ------------------------------------------------------------
//  MEMBER FUNCTION : uploadEngineering
// ------------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : incomplete
// ------------------------------------------------------------

OpsFac1Out0_A::index_type
OpsFac1Out0_A::uploadEngineering
(const double takeLoBound,                  // lower input (typically zero)
 const double takeHiBound)                  // upper input
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // PREAMBLE

  // create and fill a label object
  Label lab(d_label);                       // entity identifier

  // create a stand-in specific costs cost set
  const CostSet zeroSpecCosts(0.0);

  // INTEGRITY CHECKS

  // very basic integrity checking
  if ( takeLoBound < 0.0 )
    {
      s_logger->repx(logga::warn, "aphysical factor bound, lo", takeLoBound);
    }
  if ( takeLoBound > takeHiBound )
    {
      std::ostringstream oss;
      oss << takeLoBound << " : " << takeHiBound;
      s_logger->repx(logga::warn, "aphysical factor bounds, lo : hi", oss.str());
    }

  // EXPOSED VARIABLES

  // create one variable covering input (and thus the only choice
  // on offer for duty) -- the zero specific costs can later be
  // rectified by 'uploadShortrunCosts'
  const int inCol   = pushObj(zeroSpecCosts, lab.str("factor"));

  // load the local index into a 1-tuple
```

```
  d_cols            = boost::make_tuple(inCol);

  // FACTOR BALANCE

  // create an input balance
  const int inRow   = pushRhs(0.0, svif::E, lab.str("factor-bal"));

  // add the exposed variable
  d_cofCount        = pushCof(inRow, inCol, -1.0);

  // SINK BLOCKETTE

  // note that the sink blockette is the sole blockette in this
  // OSP -- now create an out variable for the sink blockette
  const int sinkCol = pushObj(zeroSpecCosts, lab.str("sink-in"));

  // connect to the input balance
  d_cofCount        = pushCof(inRow, sinkCol, +1.0);

  // define the input/output equality for the source blockette
  // (not required)

  // define the bounds for the sink blockette
  const int loRow   = pushRhs(takeLoBound, svif::G, lab.str("lo-bound"));
  d_cofCount        = pushCof(loRow, sinkCol, +1.0);
  const int hiRow   = pushRhs(takeHiBound, svif::L, lab.str("hi-bound"));
  d_cofCount        = pushCof(hiRow, sinkCol, +1.0);

  // YEEK 24 CODE (set by '--yeek')
  if ( xeona::yeek == 24 || xeona::yeek == 1 )
    {
      const std::string func = __func__;     // a preprocessor macro
      reportBuildIntegrity(func + " (lo/hi take call)");
    }

  // GLOBALIZE AND RETURN

  // note that external usage requires global indexing
  return globalcols(d_cols);

} // function 'uploadEngineering'

// DOWNLOAD SOLUTION CALLS

// ------------------------------------------------------------
//  MEMBER FUNCTION : downloadSolution
// ------------------------------------------------------------
//  Description  : solution recovery
//  Role         : host call
//  Techniques   : 'boost::make_tuple'
//  Status       : complete
// ------------------------------------------------------------

OpsFac1Out0_A::results_type
OpsFac1Out0_A::downloadSolution() const
{
  return boost::make_tuple(downloadVar(d_cols.get<0>()));
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : optprob.cc
//  file-create-date : Fri 17-Oct-2008 08:20 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : optimization sub-problem (OSP) and key sub-classes / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/optprob.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "optprob.h"          // companion header for this file (place first)

#include "../f/ospinfo.h"     // domain mode interpretation
#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
#include "../c/util2.h"       // free functions which offer general utilities 2
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../c/label.h"       // helper class to format solver labels

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <typeinfo>           // run-time type info, NOTE: passive reporting role only

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting

//  CODE

// ----------------------------------------------------------
//  CLASS           : OptimSubProb
// ----------------------------------------------------------
//  Description  : sub-entity-local abstraction to the domain-local solver interface
//  Role         : abstract base class for classes 'Operations' and 'Control'
//  Techniques   : "push loading" (see below), sub-classes also use 'boost::tuple'
//  Status       : complete (more or less)
//
//  Design notes
//
//      Provides for intuitive usage for module developers
//
//          Instances of this class are held by 'TechnicalAsset'
//          and 'AssetOperator' sub-classes (and perhaps others).
//
//          One purpose of this class is to shield the host
```

```
//          entity from the underlying derived OSP (optimization
//          sub-problem) and its internal formulation.  Moreover,
//          meaningful and (hopefully) consistent function names
//          can be used for the host-related calls.
//
//      Exclusive solver interface access assumed but not enforced
//
//          The solver interface is held in the instance variable
//          'd_solver' of type 'shared_ptr<svif::SolverIf>',
//          passed in at construction time.
//
//          The design assumes that each 'OptimSubProb' has
//          exclusive access to the underlying 'svif::SolverIf'
//          solver interface object during the loading of its
//          sub-problem (conversely, solution recovery is
//          read-only).  However, no attempt is made here to use
//          resource locks to prevent parallel access.  Rather
//          the underlying program-wide thread of control should
//          provide for the orderly use of the solver (otherwise
//          there will be major problems elsewhere as well).
//
//          Furthermore, it is assumed that a solver is reset at
//          the appropriate time.  This action, however, is not
//          the responsibility of this class.
//
//      Push loading of the optimization sub-problem
//
//          The constraint and variable loading calls do NOT take
//          row and col indexes respectively, but rather use and
//          return the next available index.  All such calls
//          begin with the phrase "push".
//
//      Commitment modes and subsequent action
//
//          The various commitment modes -- encoded by the
//          enum 'xeona::DomainMode' -- require different
//          objective function loading processes.
//
//          Where feasible, the correct behavior is achieved
//          using polymorphism.  However, in some cases, it is
//          simpler and cleaner to switch on the value of the
//          'd_commitmentMode' flag.
//
//          In terms of the inheritance hierarchy, operations
//          pertains to assets and control pertains to operators
//          (hopefully not as confusing as it looks).
//
//          The documentation for this class is not the right
//          place to discuss the details of the various
//          commitment modes and how the objective function is
//          actually loaded.
//
//          Note however, that the solver only ever receives one
//          set of objective function values and that, once
//          loaded, these values are not modified.  Therefore any
//          pre or post "manipulation" of object function values
//          need to be undertaken in this class.
//
//      Coupling of the operations and control (or tariff) OSPs
//
//          Coupling means adding an equality constraint to
//          connect the operations duty col (or operations sale
//          col) and the control duty col (or tariff sale col).
//          Defining duty/sale and making this coupling is the
//          responsibility of the operator host and NOT the
//          various OSPs.
//
//          The most direct way to couple is via the free
//          function:
//
//              couple(shared_ptr<svif::SolverIf> solver,
//                     const int golA,
//                     const int golB,
//                     const std::string tag)
//
//      Constraint sense for 'pushRhs' calls.
//
//          This enum is defined in the solver interface class
//          'svif::SolverIf'.
//
//      Background information on nomenclature
```

```
//
//          item                        short-name   formal description
//          ..........................................................
//          variable value              variable     primal structural variable
//          variable reduced cost       slack        dual structural variable
//          constraint value                         primal auxiliary variable
//          constraint reduced cost     shadow       dual auxiliary variable
//          ..........................................................
//
//  CAUTION: 'std::vector' containers and memory reallocation
//
//      STL vectors automatically reshuffle their internal memory
//      when their current allocation becomes insufficient.  This
//      process duly invalidates all references, pointers, and/or
//      iterators for elements of the given vector.  Josuttis
//      (1999 p149-150) discusses the associated issues in some
//      detail.  This code does not rely on any of the above
//      mechanisms and normal 'std::vector' capacity increments
//      should occur without problem.
//
// -------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger OptimSubProb::s_logger = logga::ptrLogStream();

// CREATORS

// -------------------------------------------------------
//  MEMBER FUNCTION : OptimSubProb (zero-argument)
// -------------------------------------------------------
//  Description  : zero-argument constructor
//  Role         : indirectly used in the class 'Interface' constructor
//  Techniques   : (nothing special)
//  Status       : complete
// -------------------------------------------------------

OptimSubProb::OptimSubProb() :
  d_label(),
  d_ospDesc(),

  d_rowStart(),
  d_colStart(),
  d_rowCount(),
  d_colCount(),
  d_cofCount(),
  d_colTrack(),
  d_rowTrack(),

  d_finObjs(),                              // empty vector
  d_ghgObjs(),
  d_noxObjs(),
  d_depObjs(),
  d_lucObjs(),
  d_bidObjs(),
  d_mitObjs(),
  d_nulObjs(),

  d_commitmentMode(),                       // xeona enum
  d_solver()                                // empty shared pointer
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call, zero-argument", "");
}

// -------------------------------------------------------
//  MEMBER FUNCTION : OptimSubProb (proper)
// -------------------------------------------------------

OptimSubProb::OptimSubProb
(shared_ptr<svif::SolverIf> solver,         // solver interface object
 const xeona::DomainMode    commitmentMode,
 const std::string&         ospDesc) :

  d_label(""),                              // default label is set here
  d_ospDesc(ospDesc),

  d_rowStart(solver->getConCount()),        // global indexing and also 'const'-qualified
  d_colStart(solver->getVarCount()),
  d_rowCount(0),                            // local indexing and changeable
  d_colCount(0),
```

```
  d_cofCount(0),                            // number of nonzero structural coeffs
  d_colTrack(0),                            // provided for ad-hoc use
  d_rowTrack(0),

  d_finObjs(),                              // empty vector
  d_ghgObjs(),
  d_noxObjs(),
  d_depObjs(),
  d_lucObjs(),
  d_bidObjs(),
  d_mitObjs(),
  d_nulObjs(),

  d_commitmentMode(commitmentMode),         // xeona enum
  d_solver(solver)                          // smart pointer to 'xeona::SolverIf' object
{
  // logging stringstream
  std::ostringstream put;

  // headline reporting
  static int yeekLoop          = 0;
  static const std::string tag = "OSP-";
  put << "      ";
  put << tag << std::setw(2) << std::setfill('0') << ++yeekLoop << "  ";
  put << "optimization sub-problem establishment" << "\n";
  s_logger->putx(logga::adhc, put);

  // initial reporting
  s_logger->repx(logga::xtra, "constructor call, commitment mode", d_commitmentMode);

  // confirm the commitment mode is legitimate
  if ( d_commitmentMode == xeona::e_modeNotSpecified)
    {
      // assume this setting is intentional, but comment anyway
      s_logger->repx(logga::xtra, "unspecified commitment mode in use", d_commitmentMode);
    }
  else
    {
      const tribool okay = xeona::isTwoContained(d_commitmentMode,
                                                 xeona::e_commitmentModes);
      if ( okay )                           // strictly 'true'
        {
        }
      else if ( ! okay )                    // strictly 'false'
        {
          s_logger->repx(logga::warn, "entity coding issue", okay);
        }
      else                                  // otherwise 'indeterminate'
        {
          s_logger->repx(logga::warn, "unexpected commitment mode problem", okay);
        }
    }

  // CAUTION: add a zero "shift" to the various objective vectors
  // -- necessary to keep the indexing correct
  d_finObjs.push_back(0.0);
  d_ghgObjs.push_back(0.0);
  d_noxObjs.push_back(0.0);
  d_depObjs.push_back(0.0);
  d_lucObjs.push_back(0.0);

  // low priority reporting
  s_logger->repx(logga::adhc, "low priority reporting follows", "");
  put << "  OSP row and col settings"           << "\n";
  put << "  solver address : " << d_solver      << "\n";
  if ( ! d_label.empty()   ) put << "  optional label : " << d_label   << "\n";
  if ( ! d_ospDesc.empty() ) put << "  description : "    << d_ospDesc << "\n";
  put << "      row count  : " << d_rowCount    << "\n"
      << "      row offset : " << d_rowStart    << "\n"
      << "      col count  : " << d_colCount    << "\n"
      << "      col offset : " << d_colStart    << "\n";
  s_logger->putx(logga::adhc, put);

} // proper constructor

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~OptimSubProb
// ----------------------------------------------------------

OptimSubProb::~OptimSubProb()               // destructor definition is mandatory
{
```

```
  if ( d_label.empty() )
    {
      s_logger->repx(logga::xtra, "destructor call, no OSP label", "");
    }
  else
    {
      // label could be simply "(not set)", see constructor code
      s_logger->repx(logga::adhc, "destructor call, OSP label", d_label);
    }
}

// HOST-RELATED CALLS - used by assets and operators

  // ------------------------------------------------------
  //  MEMBER FUNCTION : loadOspLabel
  // ------------------------------------------------------

std::string
OptimSubProb::loadOspLabel                    // optional OSP labeling
(const std::string label)                     // sub-problem (and not solver) label
{
  const std::string previous = d_label;
  d_label                    = label;
  s_logger->repx(logga::xtra, "resetting OSP label, now", d_label);
  s_logger->repx(logga::adhc, "resetting OSP label, was", previous);
  return previous;
}

//  CALLS PROVIDED FOR TEST PURPOSES

// ------------------------------------------------------
//  MEMBER FUNCTION : reportBuildIntegrity
// ------------------------------------------------------

bool                                          // should always be true
OptimSubProb::reportBuildIntegrity            // can call any time
(const std::string comment)                   // note default value
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // start and count checks
  bool ret = true;                            // presume build integrity at outset
  int gowCountSolver = d_solver->getConCount();
  int golCountSolver = d_solver->getVarCount();
  if ( gowCountSolver != d_rowStart + d_rowCount ) ret = false;
  if ( golCountSolver != d_colStart + d_colCount ) ret = false;

  // set integrity string
  const std::string okay
    = ret ? "start and count checks passed" : "start and count checks failed";

  // run-time type information (expensive)
  const std::string subtype = xeona::demangle(typeid(*this).name());  // CAUTION: deref

  // process the commitment mode
  std::string sCommitmentMode          = "(not overwritten)";
  const std::vector<std::string> cmode = xeona::infoDomainModeLong(d_commitmentMode);
  if ( cmode.size() == 1 )
    {
      sCommitmentMode = cmode.at(0);              // indirectly 'DomainModeInfo::longform'
      boost::algorithm::trim(sCommitmentMode);    // remove special alignment
    }
  else
    {
      const std::string sep = " ";
      std::ostringstream oss;
      oss << d_commitmentMode << " = " << xeona::reducedVector(d_commitmentMode, sep);
      sCommitmentMode = oss.str();
      ret = false;
    }

  std::ostringstream buf;
  buf << "  OSP build integrity report"                                    << "\n";
  if ( ! comment.empty() ) buf << "    call-time comment   : " << comment    << "\n";
  buf << "    OSP optional label  : " << d_label                           << "\n"
      << "    OSP sub-class       : " << subtype                           << "\n"
      << "    build integrity     : " << okay                              << "\n"
      << "    commitment mode     : " << sCommitmentMode                   << "\n"
      << "    solver gow count    : " << gowCountSolver                    << "\n"
      << "    OSP row start       : " << d_rowStart                        << "\n"
      << "    OSP row count       : " << d_rowCount                        << "\n"
```

```
      << "    solver gol count    : " << golCountSolver                        << "\n"
      << "    OSP col start       : " << d_colStart                            << "\n"
      << "    OSP col count       : " << d_colCount                            << "\n"
      << "    OSP non-zero coeffs : " << d_cofCount                            << "\n"
      << "    local summary       : " << d_rowCount << " x " << d_colCount     << "\n";

  if ( ret == true )
    {
      std::ostringstream put;
      put << buf.str();
      s_logger->putx(logga::dbug, put);
    }
  else
    {
      s_logger->repx(logga::warn, "OSP build failure", "");
      std::ostringstream put;
      put << "  optimization sub-problem build integrity FAILURE" << "\n"
          << buf.str();
      s_logger->putx(logga::info, put);
    }
  return ret;
}

// PUSH CALLS (and similar)

// ------------------------------------------------------------
//  MEMBER FUNCTION : pushIncShift (dual documentation)
// ------------------------------------------------------------
//  Description  : push increment the objective "shift"
//  Role         : problem building by sub-classes
//  Techniques   : virtual functions with substantive definitions, 'd_commitmentMode'
//  Status       : near-complete
//
//  Design notes
//
//      The "shift" is the constant term in the objective
//      function.  The "shift" does not influence short-run or
//      other optimization, but it does get added to the final
//      objective value.
//
//      The 'double' variant if for non-short-run commitment
//      modes, in contrast to the 'CostSet' variant.
//
//      At the time of writing, the two 'repushShift'
//      non-incremental "shift" value overwriting functions are
//      not provided but could be easily added.
//
//  CAUTION: which objectives to implement
//
//      See below for a caution and warning regarding coding
//      design.
//
// ------------------------------------------------------------

// DOUBLE VARIANT

void                                        // in effect, col zero
OptimSubProb::pushIncShift                  // increment objective function constant term
(const double        shiftValue,            // non-short-run commitment mode
 const CostSet&      specCosts)
{
  // CAUTION: it remain an open question as to whether the
  // 'specCosts' argument should increment the various general
  // objective vectors -- nonetheless, as at commit r5226
  // (30-Sep-2010), there are no calls to this function

  // emit a warning
  s_logger->repx(logga::warn, "unresolved coding design issue", "");
  std::ostringstream put1;
  put1 << "  this function increments the shifts for both the selected"          << "/n"
       << "  commitment objective vector and the various general objective"       << "\n"
       << "  vectors"                                                            << "\n"
       << ""                                                                     << "\n"
       << "  the question of whether it should do the latter remains unresolved"  << "\n"
       << ""                                                                     << "\n"
       << "  if you see this message, fix the issue"                             << "\n";
  if ( specCosts.isZero() )
    {
      put1 << ""                                                                 << "\n"
           << "  but note in this particular case the general increment is zero " << "\n";
    }
```

```
  s_logger->putx(logga::warn, put1);

  // increment the various objective vectors as required
  d_finObjs.at(0) += specCosts.fin;
  d_ghgObjs.at(0) += specCosts.ghg;
  d_noxObjs.at(0) += specCosts.nox;
  d_depObjs.at(0) += specCosts.dep;
  d_lucObjs.at(0) += specCosts.luc;

  // increment the various objective vectors as required
  switch ( d_commitmentMode )
    {
    case xeona::e_auctionLmp: d_bidObjs.at(0) += shiftValue; break;
    case xeona::e_adminMerit: d_mitObjs.at(0) += shiftValue; break;
    case xeona::e_adminFirst: d_nulObjs.at(0) += shiftValue; break;
    case xeona::e_shortrunFin:
    case xeona::e_shortrunGhg:
    case xeona::e_shortrunNox:
    case xeona::e_shortrunDep:
    case xeona::e_shortrunLuc:
      {
        s_logger->repx(logga::warn, "wrong call, entity design issue", "");
        std::ostringstream put2;
        put2 << "  should be passing a 'CostSet' and not a 'double' coefficient" << "\n";
        s_logger->putx(logga::dbug, put2);
      }
      break;
    default:
      std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
      break;
    }

  // modify the solver
  d_solver->incShift(shiftValue);
}

// COSTSET VARIANT

void                                        // in effect, col zero
OptimSubProb::pushIncShift                  // increment objective function constant term
(const CostSet&      shiftIncValues)        // all commitment modes
{
  // increment the various objective vectors as required
  d_finObjs.at(0) += shiftIncValues.fin;
  d_ghgObjs.at(0) += shiftIncValues.ghg;
  d_noxObjs.at(0) += shiftIncValues.nox;
  d_depObjs.at(0) += shiftIncValues.dep;
  d_lucObjs.at(0) += shiftIncValues.luc;

  // recover the required "shift" value
  double shiftval = 0.0;
  switch ( d_commitmentMode )
    {
    case xeona::e_shortrunFin: shiftval = shiftIncValues.fin; break;
    case xeona::e_shortrunGhg: shiftval = shiftIncValues.ghg; break;
    case xeona::e_shortrunNox: shiftval = shiftIncValues.nox; break;
    case xeona::e_shortrunDep: shiftval = shiftIncValues.dep; break;
    case xeona::e_shortrunLuc: shiftval = shiftIncValues.luc; break;
    case xeona::e_auctionLmp:
    case xeona::e_adminMerit:
    case xeona::e_adminFirst:
      {
        s_logger->repx(logga::adhc, "details below (call seems correct)", "");
        std::vector<std::string> modes = xeona::infoDomainModeLong(d_commitmentMode);
        std::string mode = "(some problem)";           // default value
        if ( modes.size() == 1 ) mode = modes.front(); // unique mode encountered
        std::ostringstream put;
        put << "  call details" << "\n"
            << "    function        : " << __func__ << " with CostSet"     << "\n"
            << "    commitment mode : " << mode                            << "\n"
            << "    value           : " << shiftIncValues                  << "\n";
        // the 'specCosts' supports streaming
        s_logger->putx(logga::adhc, put);
      }
      break;
    default:
      std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
      break;
    }

  // modify the solver
```

```
  d_solver->incShift(shiftval);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : pushObj (dual documentation)
// ----------------------------------------------------------
//  Description  : push load objective function values
//  Role         : problem building by sub-classes
//  Techniques   : virtual functions with substantive definitions, 'd_commitmentMode'
//  Status       : near-complete
//
//  Design notes
//
//      The 'double' variant if for non-short-run commitment
//      modes, in contrast to the 'CostSet' variant.
//
// ----------------------------------------------------------

// DOUBLE VARIANT

const int                                   // the col index just employed
OptimSubProb::pushObj
(const double      objValue,
 const CostSet&    specCosts,
 const std::string tag)                     // defaults to empty string
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "double version");

  // load the various objective vectors
  d_finObjs.push_back(specCosts.fin);
  d_ghgObjs.push_back(specCosts.ghg);
  d_noxObjs.push_back(specCosts.nox);
  d_depObjs.push_back(specCosts.dep);
  d_lucObjs.push_back(specCosts.luc);

  // add another element as required
  switch ( d_commitmentMode )
    {
    case xeona::e_auctionLmp: d_bidObjs.push_back(objValue); break;
    case xeona::e_adminMerit: d_mitObjs.push_back(objValue); break;
    case xeona::e_adminFirst: d_nulObjs.push_back(objValue); break;
    case xeona::e_shortrunFin:
    case xeona::e_shortrunGhg:
    case xeona::e_shortrunNox:
    case xeona::e_shortrunDep:
    case xeona::e_shortrunLuc:
      {
        s_logger->repx(logga::warn, "wrong call, entity design issue", "");
        std::ostringstream put;
        put << "  should be passing a 'CostSet' and not a 'double' coefficient" << "\n";
        s_logger->putx(logga::dbug, put);
      }
      break;
    default:
      s_logger->repx(logga::warn, "coding error 03, commitment mode", d_commitmentMode);
      std::clog << "** coding error 03 in source file " << __FILE__ << std::endl;
      break;
    }

  // load the solver and return col index
  ++d_colCount;
  d_solver->loadObj(globalcol(d_colCount), objValue, tag);
  return d_colCount;
}

// COSTSET VARIANT

const int                                   // the col index just employed
OptimSubProb::pushObj
(const CostSet&    specCosts,
 const std::string tag)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "CostSet version");

  // load the various objective vectors
  d_finObjs.push_back(specCosts.fin);
  d_ghgObjs.push_back(specCosts.ghg);
  d_noxObjs.push_back(specCosts.nox);
  d_depObjs.push_back(specCosts.dep);
```

```
  d_lucObjs.push_back(specCosts.luc);

  // recover the correct objective value
  double objval = 0.0;
  switch ( d_commitmentMode )
    {
    case xeona::e_shortrunFin: objval = specCosts.fin; break;
    case xeona::e_shortrunGhg: objval = specCosts.ghg; break;
    case xeona::e_shortrunNox: objval = specCosts.nox; break;
    case xeona::e_shortrunDep: objval = specCosts.dep; break;
    case xeona::e_shortrunLuc: objval = specCosts.luc; break;
    case xeona::e_auctionLmp:
    case xeona::e_adminMerit:
    case xeona::e_adminFirst:
      {
        s_logger->repx(logga::adhc, "cost set obj load omitted", "");
        s_logger->repx(logga::adhc, "details below (call seems correct)", "");
        std::vector<std::string> modes = xeona::infoDomainModeLong(d_commitmentMode);
        std::string mode = "(some problem)";           // default value
        if ( modes.size() == 1 ) mode = modes.front(); // unique mode encountered
        std::ostringstream put;
        put << "  call details" << "\n"
            << "    function        : " << __func__  << " with CostSet"    << "\n"
            << "    commitment mode : " << mode                            << "\n"
            << "    tag             : " << tag                             << "\n"
            << "    value           : " << specCosts                       << "\n";
        // the 'specCosts' supports streaming
        s_logger->putx(logga::adhc, put);
      }
      break;
    default:
      {
        std::clog << "** coding error 04 in source file " << __FILE__ << std::endl;
        break;
      }
    }

  // load the solver and return col index
  ++d_colCount;
  d_solver->loadObj(globalcol(d_colCount), objval, tag);
  return d_colCount;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : repushObj (dual documentation)
// ------------------------------------------------------------
//  Description  : repush load objective function values
//  Role         : problem building by sub-classes
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//       Two complementary forms
//
//           The 'CostSet' form is for least-cost commitment
//           modes, whereas the 'double' form is for the other
//           modes.  Incorrect invocation triggers an warning.
//
//      Subsequent use
//
//          These two 'repushObj' functions are intended for
//          subsequent use only -- meaning that 'pushObj' should
//          have already been called.
//
// ------------------------------------------------------------

// DOUBLE VARIANT

const double                                // return the previous value, now overwritten
OptimSubProb::repushObj
(const int         col,
 const double      objValue,
 const CostSet&    specCosts)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "subsequent use only");

  // integrity checks
  if ( col <= 0 )                           // most probably zero
    {
```

```
      s_logger->repx(logga::warn, "requested col zero or less", col);
    }
  else if ( col > d_colCount )
    {
      std::ostringstream oss;
      oss << col << " : " << d_colCount;
      s_logger->repx(logga::warn, "requested col exceeds current count", oss.str());
    }

  // update the various objective vectors
  d_finObjs.at(col) = specCosts.fin;
  d_ghgObjs.at(col) = specCosts.ghg;
  d_noxObjs.at(col) = specCosts.nox;
  d_depObjs.at(col) = specCosts.dep;
  d_lucObjs.at(col) = specCosts.luc;

  // remove and replace nominated element as required
  switch ( d_commitmentMode )
    {
    case xeona::e_auctionLmp: d_bidObjs.at(col) = objValue; break;
    case xeona::e_adminMerit: d_mitObjs.at(col) = objValue; break;
    case xeona::e_adminFirst: d_nulObjs.at(col) = objValue; break;
    case xeona::e_shortrunFin:
    case xeona::e_shortrunGhg:
    case xeona::e_shortrunNox:
    case xeona::e_shortrunDep:
    case xeona::e_shortrunLuc:
      {
        s_logger->repx(logga::warn, "wrong call, entity design issue", "");
        std::ostringstream put;
        put << "  should be passing a 'double' and not a 'CostSet' coefficient" << "\n";
        s_logger->putx(logga::dbug, put);
      }
      break;
    default:
      std::clog << "** coding error 05 in source file " << __FILE__ << std::endl;
      break;
    }

  // load the solver and return col index
  return d_solver->reviseObj(globalcol(col), objValue);
}

// COSTSET VARIANT

const double                                    // return the previous value, now overwritten
OptimSubProb::repushObj
(const int          col,
 const CostSet&     specCosts)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "subsequent use only");

  // integrity checks
  if ( col > d_colCount )
    {
      std::ostringstream oss;
      oss << col << " : " << d_colCount;
      s_logger->repx(logga::warn, "requested col exceeds current count", oss.str());
    }

  // update the various objective vectors
  d_finObjs.at(col) = specCosts.fin;
  d_ghgObjs.at(col) = specCosts.ghg;
  d_noxObjs.at(col) = specCosts.nox;
  d_depObjs.at(col) = specCosts.dep;
  d_lucObjs.at(col) = specCosts.luc;

  // recover the correct objective value
  double objval = 0.0;
  switch ( d_commitmentMode )
    {
    case xeona::e_shortrunFin: objval = specCosts.fin; break;
    case xeona::e_shortrunGhg: objval = specCosts.ghg; break;
    case xeona::e_shortrunNox: objval = specCosts.nox; break;
    case xeona::e_shortrunDep: objval = specCosts.dep; break;
    case xeona::e_shortrunLuc: objval = specCosts.luc; break;
    case xeona::e_auctionLmp:
    case xeona::e_adminMerit:
    case xeona::e_adminFirst:
      {
```

```
        s_logger->repx(logga::adhc, "cost set obj load omitted", "");
        s_logger->repx(logga::adhc, "details below (call seems correct)", "");
        std::vector<std::string> modes = xeona::infoDomainModeLong(d_commitmentMode);
        std::string mode        = "(some problem)";    // default value
        const std::string label = d_solver->getVarLabel(globalcol(col));
        if ( modes.size() == 1 ) mode = modes.front(); // unique mode encountered
        std::ostringstream put;
        put << "  call details" << "\n"
            << "    function        : " << __func__  << " with CostSet"         << "\n"
            << "    commitment mode : " << mode                                 << "\n"
            << "    col (gol)       : " << col << " (" << globalcol(col) << ")" << "\n"
            << "    label           : " << label                                << "\n"
            << "    value           : " << specCosts                            << "\n";
        // the 'specCosts' supports streaming
        s_logger->putx(logga::adhc, put);
      }
      break;
    default:
      std::clog << "** coding error 06 in source file " << __FILE__ << std::endl;
      break;
    }

  // load the solver and return prior value
  return d_solver->reviseObj(globalcol(col), objval);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : markBinary
// ----------------------------------------------------------

const int
OptimSubProb::markBinary
(const int col)
{
  s_logger->repx(logga::adhc, "entering member function", "");

  d_solver->markVarBinary(globalcol(col));
  return col;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : markInteger
// ----------------------------------------------------------

const int
OptimSubProb::markInteger
(const int col)
{
  s_logger->repx(logga::adhc, "entering member function", "");

  d_solver->markVarInteger(globalcol(col));
  return col;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : pushRhs
// ----------------------------------------------------------

const int                                   // the row index just employed
OptimSubProb::pushRhs
(const double                rhsValue,
 const svif::ConstraintSense conSense,      // {svif::L, E, G, R, N} from unit 'siglp'
 const std::string           tag)           // passed thru to solver
{
  s_logger->repx(logga::adhc, "entering member function", "");

  ++d_rowCount;                             // ratchet the row count
  d_solver->loadRhs(globalrow(d_rowCount),  // global indexing
                    rhsValue,
                    conSense,
                    tag);
  return d_rowCount;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : pushCof
// ----------------------------------------------------------
//
//      As far as the solver interface is concerned, the row and
//      col indexes need not exist.  However the initial code for
//      this function tests to confirm that these indexes do
```

```
//      exist as far as this class is concerned.
//
// ------------------------------------------------------------

const int                                       // number of local nonzero structural coeffs
OptimSubProb::pushCof
(const int    row,
 const int    col,
 const double coeffValue)
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  bool failflag = false;
  // local row check
  if ( row < 1 || row > d_rowCount )
    {
      std::ostringstream buf;
      buf << row << " outside 1 thru " << d_rowCount;
      s_logger->repx(logga::dbug, "out of bounds col index", buf.str());
      failflag = true;
    }
  // local col check
  if ( col < 1 || col > d_colCount )
    {
      std::ostringstream buf;
      buf << col << " outside 1 thru " << d_colCount;
      s_logger->repx(logga::dbug, "out of bounds col index", buf.str());
      failflag = true;
    }
  if ( failflag == true )
    {
      std::ostringstream oss;
      oss << "details above if report " << logga::dbug;
      s_logger->repx(logga::warn, "out of range indexing", oss.str());
      std::ostringstream put;
      put
        << "  must abandon structural coefficient insertion" << "\n"
        << "  else GLPK 'glp_load_matrix' may complain of \"column index out of range\""
        << " and then exit on POSIX signal 6" << "\n";
      s_logger->putx(logga::info, put);
      s_logger->addSmartBlank(logga::info);
      return 0;                                 // CAUTION: early return is essential
    }

  // active code
  if ( coeffValue != 0.0 )                      // action only nonzero values
    {
      ++d_cofCount;                             // ratchet the nonzero coefficient count
      d_solver->loadCof(globalrow(row),         // global indexing
                        globalcol(col),         // global indexing
                        coeffValue);
    }
  return d_cofCount;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : pushGof (global indexing)
// ------------------------------------------------------------

const int                                       // number of global nonzero structural coeffs
OptimSubProb::pushGof                           // that is, pushCof using global row and col
(const int    gow,
 const int    gol,
 const double coeffValue)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  bool failflag = false;
  // global row check
  if ( gow < 1 || gow > d_solver->getConCount() )
    {
      std::ostringstream buf;
      buf << gow << " outside 1 thru " << d_solver->getConCount();
      s_logger->repx(logga::dbug, "out of bounds col index", buf.str());
      failflag = true;
    }
  // global col check
  if ( gol < 1 || gol > d_solver->getVarCount() )
```

```
    {
      std::ostringstream buf;
      buf << gol << " outside 1 thru " << d_solver->getVarCount();
      s_logger->repx(logga::dbug, "out of bounds col index", buf.str());
      failflag = true;
    }
  if ( failflag == true )
    {
      std::ostringstream oss;
      oss << "details above if report " << logga::dbug;
      s_logger->repx(logga::warn, "out of range indexing", oss.str());
      std::ostringstream put;
      put
        << "  must abandon structural coefficient insertion" << "\n"
        << "  else GLPK 'glp_load_matrix' may complain of \"column index out of range\""
        << " and then exit on POSIX signal 6" << "\n";
      s_logger->putx(logga::info, put);
      s_logger->addSmartBlank(logga::info);
      return 0;                                 // CAUTION: early return is essential
    }

  // active code
  if ( coeffValue != 0.0 )                      // action only nonzero values
    {
      d_solver->loadCof(gow,                    // global indexing
                        gol,                    // global indexing
                        coeffValue);
    }
  const int conCount = d_solver->getConCount();
  return conCount;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : openBnds
// ------------------------------------------------------------

int                                             // number of bound changes {0, 1, 2}
OptimSubProb::openBnds
(const int col)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // get old bounds
  const double oldLower = d_solver->getLowerBnd(globalcol(col));
  const double oldUpper = d_solver->getUpperBnd(globalcol(col));

  // define new bounds - negative and positive flow
  const double newLower = -d_solver->getInf();
  const double newUpper = +d_solver->getInf();

  // process changes
  int ret = 2;
  if ( oldLower == newLower ) --ret;
  if ( oldUpper == newUpper ) --ret;

  // additional reporting as appropriate
  // YEEK 4 CODE (set by '--yeek')
  if ( xeona::yeek == 4 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << std::showpos;
      put << "  lower old/new : " << std::setw(4) << oldLower << " " << newLower << "\n"
          << "  upper old/new : " << std::setw(4) << oldUpper << " " << newUpper << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // active code
  d_solver->reviseBnds(globalcol(col), newLower, newUpper);

  // return
  return ret;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : changeBnds
// ------------------------------------------------------------

int                                             // number of bound changes {0, 1, 2}
OptimSubProb::changeBnds
```

```
(const int    col,
 const double lowerBnd,
 const double upperBnd)
{
  // initial reporting
  s_logger->repx(logga::info, "entering member function", "");

  // get old bounds
  const double oldLowerBnd = d_solver->getLowerBnd(globalcol(col));
  const double oldUpperBnd = d_solver->getUpperBnd(globalcol(col));

  // process changes - user defined
  int ret = 2;
  if ( oldLowerBnd == lowerBnd ) --ret;
  if ( oldUpperBnd == upperBnd ) --ret;

  // active code
  d_solver->reviseBnds(globalcol(col), lowerBnd, upperBnd);

  // return
  return ret;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : resetBnds
// ------------------------------------------------------------

void
OptimSubProb::resetBnds                       // reset to defaults, see unit 'd/siglpk'
(const int col)
{
  // initial reporting
  s_logger->repx(logga::info, "entering member function", "");

  // active code
  d_solver->resetDefaultBnds(globalcol(col));     // wrapper to 'utilSetDefaultBnds'
}

// CALLS WHICH TRANSLATE OSP INDEXES

// ------------------------------------------------------------
//  MEMBER FUNCTION : globalcol
// ------------------------------------------------------------
//  Description  : convert local col index to global col index
//  Role         : enable interchange between various OSP instances holding same solver
//  Techniques   : routine
//  Status       : complete
//
//  Design notes
//
//      Some input integrity checks are also included.
//
// ------------------------------------------------------------

const int
OptimSubProb::globalcol
(const int localCol) const
{
  if ( localCol < 1 ||
       localCol > d_colCount )
    {
      std::ostringstream oss;
      oss << localCol << " outside 1 thru " << d_colCount;
      s_logger->repx(logga::warn, "out of range local col", oss.str());
      return 0;
    }
  else
    {
      return localCol + d_colStart;
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : globalrow
// ------------------------------------------------------------
//  Description  : convert local row index to global row index
//  Role         : enable interchange between various OSP instances holding same solver
//  Techniques   : routine
//  Status       : complete
// ------------------------------------------------------------
```

```
const int
OptimSubProb::globalrow
(const int localRow) const
{
  if ( localRow < 1 ||
       localRow > d_rowCount )
    {
      std::ostringstream oss;
      oss << localRow << " outside 1 thru " << d_rowCount;
      s_logger->repx(logga::warn, "out of range local row", oss.str());
      return 0;
    }
  else
    {
      return localRow + d_rowStart;
    }
}

// -------------------------------------------------------
//  MEMBER FUNCTION : localcol
// -------------------------------------------------------
//  Description  : convert global col index to local col index
//  Role         : enable interchange between various OSP instances holding same solver
//  Techniques   : routine
//  Status       : complete
// -------------------------------------------------------

const int
OptimSubProb::localcol
(const int globalCol) const
{
  const int localCol = globalCol - d_colStart;

  if ( localCol < 1 ||
       localCol > d_colCount )
    {
      std::ostringstream oss;
      oss << localCol << " outside 1 thru " << d_colCount;
      s_logger->repx(logga::warn, "out of range converted col", oss.str());
      return 0;
    }
  else
    {
      return localCol;
    }
}

// -------------------------------------------------------
//  MEMBER FUNCTION : localrow
// -------------------------------------------------------
//  Description  : convert global row index to local row index
//  Role         : enable interchange between various OSP instances holding same solver
//  Techniques   : routine
//  Status       : complete
// -------------------------------------------------------

const int
OptimSubProb::localrow
(const int globalRow) const
{
  const int localRow = globalRow - d_rowStart;

  if ( localRow < 1 ||
       localRow > d_rowCount )
    {
      std::ostringstream oss;
      oss << localRow << " outside 1 thru " << d_rowCount;
      s_logger->repx(logga::warn, "out of range converted row", oss.str());
      return 0;
    }
  else
    {
      return localRow;
    }
}

// -------------------------------------------------------
//  MEMBER FUNCTION : globalcols (1-tuple thru 9-tuple)
// -------------------------------------------------------

boost::tuple<int>
```

```
OptimSubProb::globalcols
(boost::tuple<int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()));
}

boost::tuple<int, int>
OptimSubProb::globalcols
(boost::tuple<int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()));
}

boost::tuple<int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()));
}

boost::tuple<int, int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()),
                           globalcol(localCols.get<3>()));
}

boost::tuple<int, int, int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()),
                           globalcol(localCols.get<3>()),
                           globalcol(localCols.get<4>()));
}

boost::tuple<int, int, int, int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int, int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()),
                           globalcol(localCols.get<3>()),
                           globalcol(localCols.get<4>()),
                           globalcol(localCols.get<5>()));
}

boost::tuple<int, int, int, int, int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int, int, int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()),
                           globalcol(localCols.get<3>()),
                           globalcol(localCols.get<4>()),
                           globalcol(localCols.get<5>()),
                           globalcol(localCols.get<6>()));
}

boost::tuple<int, int, int, int, int, int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int, int, int, int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()),
                           globalcol(localCols.get<3>()),
                           globalcol(localCols.get<4>()),
                           globalcol(localCols.get<5>()),
                           globalcol(localCols.get<6>()),
                           globalcol(localCols.get<7>()));
```

```
}

boost::tuple<int, int, int, int, int, int, int, int, int>
OptimSubProb::globalcols
(boost::tuple<int, int, int, int, int, int, int, int, int> localCols) const
{
  return boost::make_tuple(globalcol(localCols.get<0>()),
                           globalcol(localCols.get<1>()),
                           globalcol(localCols.get<2>()),
                           globalcol(localCols.get<3>()),
                           globalcol(localCols.get<4>()),
                           globalcol(localCols.get<5>()),
                           globalcol(localCols.get<6>()),
                           globalcol(localCols.get<7>()),
                           globalcol(localCols.get<8>()));
}

// PROBLEM GET CALLS - can be made anytime but result may not be complete

// ------------------------------------------------------------
//  MEMBER FUNCTION : getShift
// ------------------------------------------------------------

CostSet
OptimSubProb::getShift() const               // meaning the current "shift" term
{
  CostSet buf;
  buf.fin = d_finObjs.at(0);
  buf.ghg = d_ghgObjs.at(0);
  buf.nox = d_noxObjs.at(0);
  buf.dep = d_depObjs.at(0);
  buf.luc = d_lucObjs.at(0);
  return buf;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : getObjs
// ------------------------------------------------------------

std::vector<CostSet>
OptimSubProb::getObjs() const
{
  std::vector<CostSet> bufs;
  for ( unsigned i = 0;
        i          < d_finObjs.size();
        ++i)
    {
      CostSet buf;
      buf.fin = d_finObjs.at(i);
      buf.ghg = d_ghgObjs.at(i);
      buf.nox = d_noxObjs.at(i);
      buf.dep = d_depObjs.at(i);
      buf.luc = d_lucObjs.at(i);
      bufs.push_back(buf);
    }

  return bufs;
}

// SOLUTION RECOVERY CALLS (note individual and vector versions)

// ------------------------------------------------------------
//  MEMBER FUNCTION : downloadVar
// ------------------------------------------------------------

double
OptimSubProb::downloadVar                    // col primal value for underlying LP
(const int col) const
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
      return 0.0;
    }

  return d_solver->getVarValue(globalcol(col));
}
```

```
// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadVars
// -----------------------------------------------------------

std::vector<double>
OptimSubProb::downloadVars() const
{
  s_logger->repx(logga::adhc, "entering member function", "");

  std::vector<double> varVals;               // empty vector

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
      return varVals;
    }

  // active code
  varVals.push_back(0.0);                    // index 0 is simply a placeholder
  for ( int col  = 1;
        col     <= d_colCount;
        ++col )
    {
      varVals.push_back(downloadVar(col));   // see above for 'downloadVar'
    }
  return varVals;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadVarCosts
// -----------------------------------------------------------

const bool
OptimSubProb::downloadVarCosts
(CostSet& varCosts) const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
      varCosts.reset(0.0);                   // overwrite current values
      return false;
    }

  // calculations
  std::vector<double> vars = downloadVars();
  varCosts.fin = xeona::coeffProduct(d_finObjs, vars);
  varCosts.ghg = xeona::coeffProduct(d_ghgObjs, vars);
  varCosts.nox = xeona::coeffProduct(d_noxObjs, vars);
  varCosts.dep = xeona::coeffProduct(d_depObjs, vars);
  varCosts.luc = xeona::coeffProduct(d_lucObjs, vars);

  // success
  return true;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadShortrunCosts (pass-by-reference)
// -----------------------------------------------------------

const bool
OptimSubProb::downloadShortrunCosts
(CostSet& varCosts,
 CostSet& fixCosts) const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "two CostSet by ref");

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
      varCosts.reset(0.0);                   // overwrite current values
      fixCosts.reset(0.0);                   // overwrite current values
      return false;
    }
```

```
  // calculations (existing values overwritten)
  downloadVarCosts(varCosts);               // load (short-run) variable costs [1]
  fixCosts = getShift();                    // assign (short-run) fixed costs
  // [1] no need to test for success

  // success
  return true;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadShortrunCosts (tuple)
// -----------------------------------------------------------

boost::tuple
<CostSet,
 CostSet>
OptimSubProb::downloadShortrunCosts() const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "tuple CostSet return");

  // preamble
  CostSet varCosts(0.0);
  CostSet fixCosts(0.0);

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
    }
  else
    {
      // calculations (existing values overwritten)
      downloadVarCosts(varCosts);               // load (short-run) variable costs [1]
      fixCosts = getShift();                    // assign (short-run) fixed costs
      // [1] no need to test for success
    }

  // return in either case
  return boost::make_tuple(varCosts, fixCosts);
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadSlack (for LMP unit price)
// -----------------------------------------------------------

double
OptimSubProb::downloadSlack                 // row dual value for underlying LP
(const int row) const
{
  s_logger->repx(logga::adhc, "entering member function", "");

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
      return 0;
    }

  // additional reporting as appropriate
  // YEEK 7 CODE (set by '--yeek')
  if ( xeona::yeek == 7 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  slack call" << "\n"
          << "    local row  : " << row                                  << "\n"
          << "    global row : " << globalrow(row)                       << "\n"
          << "    value      : " << d_solver->getSlackValue(globalrow(row)) << "\n";
      s_logger->putx(logga::yeek, put);      // 'yeek' is acceptable in this case
    }

  // active code
  return d_solver->getSlackValue(globalrow(row));
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : downloadSlacks
// -----------------------------------------------------------

std::vector<double>
OptimSubProb::downloadSlacks() const         // slack values
```

```
{
  s_logger->repx(logga::adhc, "entering member function", "");

  std::vector<double> slackVals;

  // defensive programming
  if ( ! d_solver->isUsableSoln() )          // confirm usable solution
    {
      s_logger->repx(logga::warn, "no usable solution", "debug the call order");
      return slackVals;
    }

  // active code
  for ( int row  = 1;
        row     <= d_rowCount;
        ++row )
    {
      slackVals.push_back(downloadSlack(row));    // see above for 'downloadSlack'
    }
  return slackVals;
}

// INSTANCE AND STATIC RESET CALLS
// PONDER: which of these reset calls are needed?

// ------------------------------------------------------------
//  MEMBER FUNCTION : resetSolver
// ------------------------------------------------------------

void
OptimSubProb::resetSolver                    // will affect all other solver users
(const std::string label)                    // solver label, note default
{
  d_solver->resetProblem();
  if ( ! label.empty() )
    {
      d_solver->setProblemLabel(label);      // otherwise retain preexisting label
    }
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : resetSolver (static)
// ------------------------------------------------------------

void
OptimSubProb::resetSolver                    // will affect all other solver users
(shared_ptr<svif::SolverIf> solver,
 const std::string          label)           // solver label, note default
{
  solver->resetProblem();
  if ( ! label.empty() )
    {
      solver->setProblemLabel(label);        // relabel as appropriate
    }
}

// ------------------------------------------------------------
//  CLASS           : CouplingOsp
// ------------------------------------------------------------
//  Description  : concrete class for "coupling" two OSPs
//  Role         : used in 'constrain' calls from 'AssetOperator' instances and similar
//  Techniques   : concrete
//  Status       : complete
// ------------------------------------------------------------

// CREATORS

CouplingOsp::CouplingOsp
(shared_ptr<svif::SolverIf> solver) :
  OptimSubProb(solver,
               xeona::e_modeNotSpecified,
               "coupling")
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

CouplingOsp::~CouplingOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}
```

```
// ----------------------------------------------------------
//  MEMBER FUNCTION : coupleGols
// ----------------------------------------------------------
//  Description  : adds coupling constraint row and structural coefficients
//  Role         : called by asset operator 'constrain' function
//  Techniques   : solver interface calls
//  Status       : complete
// ----------------------------------------------------------

bool                                        // 'true' if coupling constraint added
CouplingOsp::coupleGols
(const int         golA,
 const int         golB,
 const std::string tag)                     // note default
{
  // initial reporting
  std::ostringstream oss;
  oss << golA << " " << golB;
  s_logger->repx(logga::xtra, "entering member function, gols", oss.str());

  // integrity checks
  bool okay = true;                         // presumption of data integrity
  if ( golA == golB ) okay = false;
  if ( golA <= 0 )    okay = false;
  if ( golB <= 0 )    okay = false;
  if ( okay == false )
    {
      std::ostringstream oss;
      oss << golA << " : " << golB;
      s_logger->repx(logga::warn, "problematic global col values", oss.str());
      s_logger->repx(logga::dbug, "early return", "failure");
      return false;
    }

  // process the solver contribution
  Label label(tag);
  label << "osp-coupling";

  int cofCount  = 0;                        // not actually used for anything
  const int row = pushRhs(0.0, svif::E, label.str());
  const int gow = globalrow(row);
  cofCount      = pushGof(gow, golA, +1.0);
  cofCount      = pushGof(gow, golB, -1.0);
  return true;
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::couple
// ----------------------------------------------------------
//  Description  : concrete class for "coupling" two OSPs
//  Role         : used in 'constrain' calls from 'AssetOperator' instances and similar
//  Techniques   : free function wrapper to 'Coupling::coupleGols'
//  Status       : complete
// ----------------------------------------------------------

namespace xeona
{
  bool                                      // 'true' if coupling constraint added
  couple
  (shared_ptr<svif::SolverIf> solver,
   const int                  golA,         // usually from { ops }
   const int                  golB,         // usually from { ctl con }
   const std::string          tag)          // defaults to ""
  {
    static logga::spLogger logger = logga::ptrLogStream();  // free function logger
    logger->repx(logga::xtra, "entering free function, gols", "");

    CouplingOsp couple(solver);
    return couple.coupleGols(golA, golB, tag);
  }
}

// ----------------------------------------------------------
//  CLASS           : ConnectionOsp
// ----------------------------------------------------------
//  Description  : concrete class for "connecting" block interfaces in an OSP context
//  Role         : used in 'constrain' calls from 'TechnicalAsset' instances and similar
//  Techniques   : concrete
//  Status       : complete
// ----------------------------------------------------------
```

```
// CREATORS

ConnectionOsp::ConnectionOsp() :             // zero-argument constructor
  OptimSubProb(),
  d_gol(),
  d_passCount(),
  d_lastSolverAddress("")
{
  s_logger->repx(logga::xtra, "constructor call, zero-argument", "");
}

ConnectionOsp::ConnectionOsp
(shared_ptr<svif::SolverIf> solver,
 const int                  passCount) :     // note default
  OptimSubProb(solver,
               xeona::e_modeNotSpecified,    // see unit 'f/ospmodes'
               "connection"),
  d_gol(0),
  d_passCount(passCount),
  d_lastSolverAddress("")
{
  s_logger->repx(logga::xtra, "constructor call, normal", "");
  getPassCount();                            // for its range checking side-effects
}

ConnectionOsp::~ConnectionOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// PUBLIC FUNCTIONS

// ------------------------------------------------------------
//  MEMBER FUNCTION : incPassCount
// ------------------------------------------------------------

int
ConnectionOsp::incPassCount
(shared_ptr<svif::SolverIf> solver)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code
  d_passCount++;
  const int passCount = getPassCount();      // 'getPassCount' contains range checking

  // check for solver mismatch (added to assist a bug hunt)
  std::string msg = "(not overwritten)";
  if ( checkSolverConsistency(passCount, solver, msg) )
    {
      s_logger->repx(logga::adhc, "connection OSP solver okay", msg);      // okay
    }
  else
    {
      s_logger->repx(logga::warn, "connection OSP solver issue", msg);     // problem
    }

  // return
  return passCount;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : getPassCount
// ------------------------------------------------------------

int
ConnectionOsp::getPassCount() const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, count", d_passCount);

  // active code
  if ( d_passCount < 0 )
    {
      s_logger->repx(logga::warn, "pass count strictly negative ", d_passCount);
    }
  else if ( d_passCount > 2 )
    {
      s_logger->repx(logga::warn, "pass count exceeds two", d_passCount);
    }
```

```
  return d_passCount;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : resetPassCount
// ----------------------------------------------------------

void
ConnectionOsp::resetPassCount()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, count", "");

  d_passCount = 0;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : storeGol
// ----------------------------------------------------------

void
ConnectionOsp::storeGol
(const int gol)
{
  d_gol = gol;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : recoverGol
// ----------------------------------------------------------

int
ConnectionOsp::recoverGol() const
{
  return d_gol;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : bindGols
// ----------------------------------------------------------
//  Description  : bind two global cols in order to maintain equality
//  Role         : connect interface process, couple duty process
//  Techniques   : 'svif::SolverIf' calls
//  Status       : complete - but not tested
//
//  Terminology
//
//      "gol" = global col
//      "gow" = global row
//
// ----------------------------------------------------------

bool                                        // 'true' if connection constraint added
ConnectionOsp::bindGols
(const int         gol1,
 const int         gol2,
 const std::string tag)                     // note default
{
  // initial reporting
  std::ostringstream put;
  s_logger->repx(logga::adhc, "entering member function", "");
  put << "  connection OSP pointer (me) : " << this << "\n"
      << "  given global col one        : " << gol1 << "\n"
      << "  given global col two        : " << gol2 << "\n";
  s_logger->putx(logga::adhc, put);

  // integrity checks
  if ( gol1 == gol2 )
    {
      std::ostringstream oss;
      oss << gol1 << " : " << gol2;
      s_logger->repx(logga::warn, "identical gol values", oss.str());
    }

  // integrity checks
  bool okay = true;                         // presumption of data integrity
  if ( gol1 == gol2 ) okay = false;
  if ( gol1 <= 0 )    okay = false;
  if ( gol2 <= 0 )    okay = false;
  if ( okay == false )
    {
```

```
      std::ostringstream oss;
      oss << gol1 << " : " << gol2;
      s_logger->repx(logga::warn, "problematic global col values", oss.str());
      s_logger->repx(logga::dbug, "early return", "failure");
      return false;
    }

  // process the solver contribution
  Label lab(d_label);
  lab << "iface-bal";                        // hardcoded

  int cofCount  = 0;                         // not actually used for much
  const int row = pushRhs(0.0, svif::E, lab.str());
  const int gow = globalrow(row);

  put << "  local row          : " << row  << "\n"
      << "  global row         : " << gow  << "\n"
      << "  global col storage : " << gol1 << "\n"
      << "  global col given   : " << gol2 << "\n";
  s_logger->putx(logga::adhc, put);

  cofCount      = pushGof(gow, gol1, +1.0);
  cofCount      = pushGof(gow, gol2, -1.0);

  s_logger->repx(logga::adhc, "leaving member function", "success");
  return true;                               // meaning solver row entered
}

// UTILITY FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : checkSolverConsistency
// ----------------------------------------------------------
//  Description  : check for solver consistency, no explicit effect on code logic
//  Role         : added to chase a bug involving solver consistency
//  Techniques   : shared pointer stream insertion, 'd_lastSolverAddress'
//  Status       : complete
// ----------------------------------------------------------

bool                                         // 'false' if problem detected
ConnectionOsp::checkSolverConsistency        // used by 'incPassCount'
(const int                  passCount,
 shared_ptr<svif::SolverIf> solver,
 std::string&               info)            // processing information for local use
{
  // process solver address
  std::ostringstream oss;
  oss << solver;                                    // inserts "solver.get()"
  const std::string solverAddress = oss.str();    // stringify

  // undertake integrity checks and similar as required
  switch ( passCount )
    {
    case 1:
      info                = solverAddress + " : (premature)";
      d_lastSolverAddress = solverAddress;        // update stored address string
      return true;
    case 2:
      if ( d_lastSolverAddress == solverAddress ) // string-wise comparison
        {
          info                = d_lastSolverAddress + " : " + solverAddress;
          d_lastSolverAddress = solverAddress;    // update stored address string
          return true;
        }
      else
        {
          info = d_lastSolverAddress + " : " + solverAddress;
          return false;
        }
    default:
      s_logger->repx(logga::warn, "unexpected invalid pass count", passCount);
      info = "coding problem";
      return false;
    }

} // member function 'ConnectionOsp::checkSolverConsistency'

// ----------------------------------------------------------
//  CLASS           : ControlOsp (penalty-based control)
// ----------------------------------------------------------
//  Description  : abstract class covering penalty-based control (PBC)
```

```
//  Role         : parent for concrete "Ctl" sub-classes
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//  CAUTION: code order
//
//      This definition must precede the definition for class
//      'Operations'.
//
// ----------------------------------------------------------

// CREATORS

ControlOsp::ControlOsp
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const xeona::DomainMode    commitmentModeSum,
 const std::string&         ospDesc) :
  OptimSubProb(solver, commitmentMode),
  d_dutyCol(0)
{
  s_logger->repx(logga::xtra, "constructor call", "");

  // integrity checks
  const tribool okay = xeona::isTwoContained(commitmentMode,
                                             commitmentModeSum);

  // CAUTION: the following 'boost::tribool' tests work as
  // expected, however not all constructs behave intuitively
  if      ( okay )                          // strictly 'true'
    {
      std::ostringstream oss;
      oss << commitmentMode << " two-contained in " << commitmentModeSum;
      s_logger->repx(logga::adhc, "commitment mode fits", oss.str());
    }
  else if ( ! okay )                        // strictly 'false'
    {
      std::ostringstream oss;
      oss << commitmentMode << " not two-contained in " << commitmentModeSum;
      s_logger->repx(logga::warn, "commitment mode mismatch", oss.str());
    }
  else                                      // strictly 'indeterminate'
    {
      std::ostringstream oss;
      oss << commitmentMode << " : " << commitmentModeSum;
      s_logger->repx(logga::warn, "commitment mode data issue", oss.str());
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~ControlOsp
// ----------------------------------------------------------

ControlOsp::~ControlOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// FILL PROBLEM CALLS

// ----------------------------------------------------------
//  MEMBER FUNCTION : ControlOsp::setFloorDuty
// ----------------------------------------------------------

void
ControlOsp::setFloorDuty
(const double mustRunDuty,
 const int    teasDutyCol)
{
  Label lab(d_label);
  lab << "set-duty";
  const int floorRow = pushRhs(mustRunDuty, svif::G, lab.str("floor"));
  d_cofCount         = pushCof(floorRow, teasDutyCol, +1.0);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ControlOsp::setCeilingDuty
// ----------------------------------------------------------
```

```
void
ControlOsp::setCeilingDuty
(const double maxDuty,
 const int    teasDutyCol)
{
  Label lab(d_label);
  lab << "set-duty";
  const int ceilingRow = pushRhs(maxDuty, svif::L, lab.str("ceiling"));
  d_cofCount           = pushCof(ceilingRow, teasDutyCol, +1.0);
}

// ----------------------------------------------------------
//  CLASS           : OperationsOsp
// ----------------------------------------------------------
//  Description  : abstract class covering operations
//  Role         : parent for concrete "Ops" sub-classes
//  Techniques   :
//  Status       : first-pass complete but under development
//
//  Design notes
//
//      OSP coupling
//
//          The constraint equation which couples the operations
//          and control OSPs is the first call to 'pushRhs' for
//          this class -- in other words, it will have a local
//          row index of unity.
//
//      Base class members
//
//          Note the need to scope qualify the call to
//          'OptimSubProb::pushZeroObj'.
//
//      Use of 'd_commitmentMode' flag
//
//          This function uses the 'd_commitmentMode' flag to
//          make the right choices -- and while the use of a flag
//          in this manner is counter to object-oriented design,
//          it is hard to get the information across using
//          inheritance.
//
//  CAUTION: code layout
//
//      The definition for class 'Operations' MUST precede this
//      definition.
//
// ----------------------------------------------------------

// CREATORS

OperationsOsp::OperationsOsp
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string&         ospDesc) :
  OptimSubProb(solver, commitmentMode, ospDesc)
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

OperationsOsp::~OperationsOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// PROBLEM BUILDING CALLS (and similar)

void
OperationsOsp::uploadShortrunCosts
(const CostSet& dutySpecCosts,
 const int      gol)                        // zero is valid
{
  // filter on index zero
  if ( gol == 0 )
    {
      // initial reporting
      s_logger->repx(logga::xtra, "entering member function, shift use", gol);
      s_logger->repx(logga::dbug, "shift term overwrite will occur", "");
      repushObj(gol, dutySpecCosts);         // zero means zero
    }
  else
    {
```

```
      // initial reporting
      const int col = localcol(gol);
      std::ostringstream oss;
      oss << col << " (" << gol << ")";
      s_logger->repx(logga::adhc, "entering member function, col (gol)", oss.str());
      s_logger->repx(logga::dbug, "normal overwrite will occur", "");
      repushObj(col, dutySpecCosts);     // CAUTION: local col index required
    }
}

void
OperationsOsp::uploadShortrunCosts
(const CostSet& shiftCosts)
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "no gol version");
  s_logger->repx(logga::dbug, "shift term increment will occur", "");

  // active code
  pushIncShift(shiftCosts);
}

// SOLUTION RECOVERY CALLS

const double
OperationsOsp::downloadSolnVar
(const int gol) const
{
  return downloadVar(localcol(gol));
}

const double
OperationsOsp::downloadSolnSlack
(const int gow) const
{
  return downloadSlack(localrow(gow));
}

const bool                                  // 'false' means not selected or tripped
OperationsOsp::downloadRunStatus
(const int gol) const                       // trip global column
{
  return d_solver->getBinaryVarValue(gol);
}

// ------------------------------------------------------------
//  CLASS           : QuantityOsp
// ------------------------------------------------------------

QuantityOsp::QuantityOsp
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string&         ospDesc) :
  OptimSubProb(solver, commitmentMode, ospDesc)
{
  s_logger->repx(logga::dbug, "constructor call", "");
}

QuantityOsp::~QuantityOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ------------------------------------------------------------
//  CLASS           : OfferOsp
// ------------------------------------------------------------

OfferOsp::OfferOsp
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string&         ospDesc) :
  OptimSubProb(solver, commitmentMode, ospDesc)
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

OfferOsp::~OfferOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}
```

```
// -----------------------------------------------------------
//  CLASS           : JunctionOsp
// -----------------------------------------------------------

JunctionOsp::JunctionOsp
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string&         ospDesc) :
  OptimSubProb(solver,
               commitmentMode,
               ospDesc)
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

JunctionOsp::~JunctionOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// -----------------------------------------------------------
//  CLASS           : LmpNodeOsp
// -----------------------------------------------------------

LmpNodeOsp::LmpNodeOsp
(shared_ptr<svif::SolverIf> solver,
 const xeona::DomainMode    commitmentMode,
 const std::string&         ospDesc) :
  OptimSubProb(solver,
               commitmentMode,
               ospDesc)
{
  s_logger->repx(logga::xtra, "constructor call", "");
}

LmpNodeOsp::~LmpNodeOsp()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : overseer.cc
//  file-create-date : Thu 07-Aug-2008 19:53 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : top-level overseer entity (singleton) / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/overseer.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "overseer.h"         // companion header for this file (place first)

#include "../f/trav.h"        // domain graph traversal class
#include "../f/ospmodes.h"    // domain mode enums (header only)
#include "../f/cta.h"         // captrans algorithm
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/costs.h"       // cost sets and support
#include "../b/domcon.h"      // domain controller entity
#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams
#include <vector>             // STL sequence container

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// STATIC DEFINITIONS

unsigned Overseer::s_count = 0;

// CREATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : Overseer
// ----------------------------------------------------------

Overseer::Overseer
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  FullEntity(entityId, record),
  CostRegisterOverseer(record),
  d_captrans_algorithm(record.tieSingle<std::string>("captrans-algorithm")),
  d_ranked_orig_domains(record.tieSingle<std::string>("ranked-orig-domains")),
  d_rankedOrigDomains()                       // vector
{
  // initial reporting and integrity checks
  s_logger->repx(logga::dbug, "constructor call", getIdAndKind());
```

```
  ++s_count;                                    // instance count
  if ( s_count > 1 )
    {
      s_logger->repx(logga::warn, "single overseer condition violated", s_count);
    }
  d_builtinRemark = "mandatory entity";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Overseer
// ----------------------------------------------------------

Overseer::~Overseer()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());  // singletons noisier
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : factoryInitialize
// ----------------------------------------------------------

void
Overseer::factoryInitialize()                // called by factory after construction
{
}

// SIMULATION POINT OF ENTRY

// ----------------------------------------------------------
//  MEMBER FUNCTION : run
// ----------------------------------------------------------
//  Description  : simulation point of entry
//  Role         : invoked by free function 'xeona::simulate'
//  Techniques   : string list to vector via templated 'listToVec'
//  Status       : mostly complete
//
//  Design notes
//
//      Function return values
//
//          The following return statuses are supported:
//
//              premature 'false' : on solver infeasibility
//              full-term 'false' : as above but option '--again' deployed
//              full-term 'true'  : normal exit
//
//          The '--again' command-line option means do not quit
//          on bad solver returns.
//
//      Dynamic domain graphs
//
//          The addition of some more 'TravDepthFirst' reset
//          calls would enable the domain graph to be revised
//          between intervals, that is, the domain graph would
//          become a dynamic structure.  The ranked original
//          domains would also need an updating process too if
//          this were to be fully general.  The reset calls would
//          look something like this:
//
//              dfs.reset(new TravDepthFirst(d_rankedOrigDomains))
//
//      Yet to be implemented functionality
//
//          See code for comments and stubs for future
//          development.  This functionality is not essential to
//          the basic 'xeona' model and has been omitted during
//          the first development pass.
//
// ----------------------------------------------------------

bool
Overseer::run
(const int steps)
{
  // ---------------------------------
  //  preliminaries
  // ---------------------------------

  // initial logging
  s_logger->repx(logga::info, "entering member function, steps", steps);
```

```
  // set function return value
  bool runRet = true;                       // presume success

  // prepare ranked domain controller list
  const int origDomainsCnt
    = FullEntity::listToVec<DomainController>(d_ranked_orig_domains, d_rankedOrigDomains);

  // report
  s_logger->repx(logga::adhc, "originating domains count", origDomainsCnt);

  // ---------------------------------
  //  domain graph traversal
  // ---------------------------------

  s_logger->repx(logga::adhc, "domain graph DFS calls start", "");

  // declare a depth first search object, used to invoke domain
  // functions 'establish' 'initialize' 'constrain' 'washup' and
  // 'conclude'

  shared_ptr<TravDepthFirst> dfs;           // empty shared pointer

  // the construction of a 'TravDepthFirst' object also invokes a
  // depth first search (DFS) traversal of the domain graph --
  // this object is latter passed various function pointers

  dfs.reset(new TravDepthFirst(d_rankedOrigDomains));  // domain graph also traversed

  // the search also counts the number of components in the
  // graph, whereby a component is a maximally connected
  // sub-graph

  const int componentCount = dfs->getComponentsCount();
  s_logger->repx(logga::adhc, "component count", componentCount);

  // ---------------------------------
  //  cost register reset
  // ---------------------------------

  CostRegister::resetRegister();            // reset entire register

  // ---------------------------------
  //  establish domains sweep
  // ---------------------------------

  // NOTE: cost register resets are also required for domain
  // controllers, asset operators, and technical assets as part
  // of their establishment loops -- indeed, this procedure is
  // somewhat brittle, it would be much better to have a full
  // entity tree traversal with both hollow and active reset
  // calls

  s_logger->repx(logga::adhc, "establish domains sweep starts", "");

  // CAUTION: for some unknown reason, the following two lines
  // could not be combined into one line -- so be careful!

  funcPtrVoid funcEstablishDomain = &DomainController::establishDomain;
  const int establishCount        = dfs->callDomains(funcEstablishDomain);
  s_logger->repx(logga::xtra, "establish sweep complete, domains", establishCount);

  // ---------------------------------
  //  restructure domains sweep
  // ---------------------------------

  s_logger->repx(logga::adhc, "restructure domains sweep starts", "");

  // CAUTION: for some unknown reason, the following two lines
  // could not be combined into one line -- so be careful!

  funcPtrVoid funcRestructureDomain = &DomainController::restructureDomain;
  const int restructureCount        = dfs->callDomains(funcRestructureDomain);
  s_logger->repx(logga::xtra, "restructure sweep complete, domains", restructureCount);

  // ---------------------------------
  //  step thru horizon
  // ---------------------------------

  s_logger->repx(logga::adhc, "step thru horizon starts, steps", steps);

  // CAUTION: the step count is ZERO-based
```

```
// step thru the horizon steps
for ( int step = 0; step < steps; ++step )
  {
    // ---------------------------------
    //  responsive contexts sweep
    // ---------------------------------

    // update the responsive contexts (that is, contexts
    // responsive to other contexts and events) here in due
    // course

    // ---------------------------------
    //  thermal sub-net sweep
    // ---------------------------------

    // stub for a thermal sub-net (TSN) sweep

    // ---------------------------------
    //  initialize domains sweep
    // ---------------------------------

    s_logger->repx(logga::adhc, "initialize domains sweep starts", "");

    // each domain initializes its operators, assets, and
    // selgates, while also passing down 'step' and 'solver'
    // information in the process

    // CAUTION: for some unknown reason, the following two
    // lines could not be combined into one line -- so be
    // careful!

    funcPtrStep funcInitializeDomain = &DomainController::initializeDomain;
    const int initCount              = dfs->callDomains(funcInitializeDomain, step);
    s_logger->repx(logga::xtra, "initialize sweep complete, domains", initCount);

    // ---------------------------------
    //  constrain domains sweep
    // ---------------------------------

    // code not required as constraint undertaken as part of
    // 'DomainController::capset' and
    // 'DomainController::transolve' calls, code last in r3030

    // ---------------------------------
    //  CTA preparations
    // ---------------------------------

    s_logger->repx(logga::adhc, "system CTA calls start", "");

    // the various domain and gateway flags are suitably
    // initialized on construction and are reset at the end of
    // each call

    // if the system fails to solve, successive levels of
    // constraint removal are employed -- however, supply
    // rationing is neither designed into 'xeona' nor
    // implemented

    shared_ptr<CapTransAlg> cta;           // empty shared pointer

    // create an appropriate CTA object
    if      ( d_captrans_algorithm == "fixed" )
      {
        cta.reset(new CtaFixed(step, d_rankedOrigDomains));
      }
    else if ( d_captrans_algorithm == "simple" )
      {
        cta.reset(new CtaSimple(step, d_rankedOrigDomains));
      }
    else if ( d_captrans_algorithm == "hop-relit" )
      {
        // TOFIX: hop-relit: reinstate after working up 'CtaHopRelit' in unit 'f/cta'
        // cta.reset(new CtaHopRelit(step, d_rankedOrigDomains));
      }
    else
      {
        const bool ret = false;
        s_logger->repx(logga::warn,
                       "unrecognized captrans algorithm name",
                       d_captrans_algorithm);
```

```
    s_logger->repx(logga::info, "abandoning member function, return", ret);
    return ret;
  }

// ---------------------------------
//  CTA attempts
// ---------------------------------

// attempt normal operation
if ( ! cta->captrans(xeona::e_withholdOkay) ) // indeterminate (bad data) passes
  {
    if ( xeona::again == false )       // command-line option '--again' absent
      {
        s_logger->repx(logga::info, "abandoning run (else use --again)", "");
        return false;
      }

    // then ban withholding and retry
    if ( ! cta->captrans(xeona::e_withholdBan) )
      {
        // customer complaints can be expected!
        if ( xeona::nopro == false )    // command-line option '--krazy' absent
          {
            s_logger->repx(logga::dbug, "abandoning run (else use --krazy)", "");
            return false;              // abandon simulation
          }
        else
          {
            if ( runRet == true )
              {
                s_logger->repx(logga::dbug, "return toggled to false", "");
              }
            runRet = false;            // reset return
            continue;                  // abandon step and try to continue
          }
      }
  } // top level if

// ---------------------------------
//  abandon if yeek 28
// ---------------------------------

// additional action as appropriate
// YEEK 28 CODE (set by '--yeek')
if ( xeona::yeek == 28 )
  {
    std::string msg;
    msg += "part way thru function '";
    msg += __func__;
    msg += "'";
    msg += " using withholding okay";
    s_logger->repx(logga::xtra, "will throw xeona::yeek_abandon", xeona::yeek);
    throw xeona::yeek_abandon(msg);
  }

// ---------------------------------
//  washup domains sweep
// ---------------------------------

s_logger->repx(logga::adhc, "washup domains sweep starts", "");

funcPtrVoid funcWashupDomain = &DomainController::washupDomain;
const int washupCount        = dfs->callDomains(funcWashupDomain);
s_logger->repx(logga::xtra, "washup sweep complete, domains", washupCount);

// ---------------------------------
//  cost consolidation sweep
// ---------------------------------

s_logger->repx(logga::adhc, "consolidate domains sweep starts", "");

CostSet var(0.0);
CostSet fix(0.0);
CostSet emb(0.0);
funcPtrStepCs3 funcConsolDomain = &DomainController::consolidateDomain;
const int consolCount           = dfs->callDomains(funcConsolDomain,
                                                   step,
                                                   var, fix, emb);
importCosts(step, var, fix, emb);      // load the new consolidated costs
s_logger->repx(logga::xtra, "consolidate sweep complete, domains", consolCount);
```

```
    } // steps loop

  s_logger->repx(logga::adhc, "step thru horizon complete", "");

  // ---------------------------------
  //  conclude domains sweep
  // ---------------------------------

  s_logger->repx(logga::adhc, "conclude domains sweep starts", "");

  funcPtrVoid funcConcludeDomain = &DomainController::concludeDomain;
  const int concludeCount        = dfs->callDomains(funcConcludeDomain);
  s_logger->repx(logga::xtra, "conclude sweep complete, domains", concludeCount);

  // return
  s_logger->repx(logga::adhc, "leaving member function, return", runRet);
  return runRet;

} // 'Overseer::run'

// see r2151 for 'xeona::e_crisisOperation' nesting in CTA attempts code

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : propdata.cc
//  file-create-date : Thu 05-Feb-2009 11:43 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : mass-based commodity property data / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/propdata.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "propdata.h"         // companion header for this file (place first)

#include <limits>             // numeric_limits<T>::infinity() and similar

//  DEFINITIONS: some typical MASS-specific physico-chemical property values
//  source: Wikipedia, July 2008

namespace
{
  const double TOGET = std::numeric_limits<double>::quiet_NaN();
} // unnamed namespace

namespace xeona
{
  const double massCp_steam         = +2080;      // [J/kgC]
  const double massCp_ice           = +2050;      // [J/kgC]
  const double massCp_liquidWater   = +4181;      // [J/kgC]

  const double massHhv_hardCoal     = +24e6;      // [J/kg]
  const double massHhv_lignite      = TOGET;      // [J/kg]
  const double massHhv_natGas       = +54e6;      // [J/kg]
  const double massHhv_wood         = +15e6;      // [J/kg]

  const double massCO2_hardCoal     = 1.83;       // [kg/kg]
  const double massCO2_lignite      = ::TOGET;    // [kg/kg]
  const double massCO2_natGas       = ::TOGET;    // [kg/kg]
  const double massCO2_wood         = ::TOGET;    // [kg/kg]

  const double massGhg_hardCoal     = ::TOGET;    // [kg/kg]
  const double massGhg_lignite      = ::TOGET;    // [kg/kg]
  const double massGhg_natGas       = ::TOGET;    // [kg/kg]
  const double massGhg_wood         = ::TOGET;    // [kg/kg]

  const double massGray_hardCoal    = ::TOGET;    // [kg/kg]
  const double massGray_lignite     = ::TOGET;    // [kg/kg]
  const double massGray_natGas      = ::TOGET;    // [kg/kg]
  const double massGray_wood        = ::TOGET;    // [kg/kg]
} // namespace xeona

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : registr.cc
//  file-create-date : Wed 20-Jun-2007 13:25 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : entity sub-class registrations / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/register.cc $

//  ADDING SUB-ENTITY CLASSES
//
//  To add a new sub-entity class, see the guidelines in: SUBENTITY_AUTHORING.txt

//  LOCAL AND SYSTEM INCLUDES

#include "register.h"         // companion header for this file (place first)

#include "../c/factory.h"     // entity factory
#include "../b/commods.h"     // commodities hierarchy
#include "../b/entity.h"      // entity base class

// ~~~~~ step one of two : add corresponding header ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#ifndef _XUTEST                           // unit testing code

# include "../b/builtins.h"             // builtin sub-entities
# include "../b/tests.h"                // generic test sub-entities for --inbuilt

# include "../b/overseer.h"             // top-level overseer entity (singleton)
# include "../b/domcon.h"               // domain controller entity

# include "../b/commods.h"              // commodities hierarchy
# include "../b/commods01.h"            // concrete commodities 1

# include "../b/asop01.h"               // concrete asset operators 1
# include "../b/asop02.h"               // concrete asset operators 2
# include "../b/asop03.h"               // concrete asset operators 3
# include "../b/gate01.h"               // concrete gateways 1
# include "../b/junc01.h"               // concrete demand split junctions
# include "../b/junc02.h"               // concrete demand join junctions
# include "../b/node01.h"               // concrete LMP nodes 1
# include "../b/node02.h"               // concrete LMP nodes 2
# include "../b/teas01.h"               // concrete technical assets 1
# include "../b/teas02.h"               // concrete technical assets 2
# include "../b/teas03.h"               // concrete technical assets 3
# include "../b/teas04.h"               // concrete technical assets 4
# include "../b/teasdev.h"              // concrete technical assets development

# include "../e/cxamb01.h"              // concrete ambient conditions contexts 1
# include "../e/cxamb02.h"              // concrete ambient conditions contexts 2
# include "../e/cxecon01.h"             // concrete economic contexts 1
# include "../e/cxpol01.h"              // concrete public policy contexts 1
```

```
#endif // _XUTEST

// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <cctype>             // C-style char classification, case conversion
#include <string>             // C++ strings
#include <typeinfo>           // run-time type info, NOTE: passive reporting role only
#include <utility>            // STL pair, make_pair()

//  CODE

namespace // unnamed
{
  // ----------------------------------------------------------
  //  FREE FUNCTION   : ::createType <>
  // ----------------------------------------------------------
  //  Description  : free function
  //  Role         : provide 'EntityFactory' create functions
  //  Scope        : this file only
  //  Techniques   : RTTI (run-time type identification), templates, smart pointers
  //  Status       : working
  // ----------------------------------------------------------
  //
  //  Design notes
  //
  //       This function is called from
  //       'EntityFactory::createEntityBind' after being
  //       registered earlier with explicit template
  //       instantiation.
  //
  //  CAUTION: shared pointers
  //
  //      First, a raw pointer must only be used once to create a
  //      (bunch of) shared pointers -- this is done in the call
  //      marked '#' below.
  //
  //      Second, that shared pointers to 'this' cannot be
  //      created in constructors and can only be employed after
  //      the constructor has returned.  Hence the
  //      'processFabricatedPtr' call.
  //
  // ----------------------------------------------------------

  template <typename E>                     // E is a sub-class of Entity
  shared_ptr<Entity>                        // CAUTION: <E> creates a compile-time error
  createType
  (const std::string& entityId,             // unique identifier
   Record&            r)                    // associated record
  {
    static logga::spLogger logger = logga::ptrLogStream();

    std::string compilerName;               // cleaned up type name
    compilerName = typeid(E).name();        // name() returns const char*
    compilerName = xeona::demangle(compilerName);

    logger->repx(logga::dbug, "new sub-entity, typeid(E) yields", compilerName);

    logger->repx(logga::adhc, "about to call new", entityId);
    shared_ptr<E> sp(new E(entityId, r));    // constructor call (# see caution above)

    logger->repx(logga::adhc, "about to process my shared pointer", entityId);
    sp->processFabricatedPtr(sp);           // called with implicit instantiation

    logger->repx(logga::adhc, "about to initialize", entityId);
    sp->factoryInitialize();                // factory initialization call

    return sp;
  }

} // unnamed namespace

namespace xeona
{

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::registerEntyCreators
  // ----------------------------------------------------------
  //  Description  : support for entity factory
```

```
  //  Role         : called by the appropriate function to register Entity creators
  //  Techniques   : function-like preprocessor macro
  //  Status       : working
  //
  //  Design notes
  //
  //      Protection is provided against repeat calls.  And a
  //      warning is also logged.
  //
  //      This function is invoked by the 'xeona::simulate' free
  //      function (which is, in turn, duly invoked by the main
  //      function).
  //
  // -------------------------------------------------------

  void
  registerEntyCreators()
  {
    // protection against repeat calls
    static bool firstCall = true;
    static logga::spLogger logger = logga::ptrLogStream();
    if ( ! firstCall )
      {
        logger->repx(logga::warn, "duplicate call", "check client code");
        return;
      }
    firstCall = false;
    logger->repx(logga::dbug, "first time", "");

// ~~~~~ step two of two : add corresponding creation call ~~~~~~~~~~~~~~~~~~~~~

    // 'EFIRE' is simply a programming convenience.  'EFIRE' is
    // known as a function-like preprocessor macro, see Lischner
    // (2003 p279) for further discussion.  Note that no
    // additional space is needed, somewhat surprisingly, in the
    // "<b>" definition ('cpp' must contain some smarts!).

#define EFIRE(a,b) EntityFactory::iface()->registerEntity(a, &::createType<b>)

    // note my emacs macro 'robbie-xeona-efire' : f10 S-z .. TestClass

#ifndef _XUTEST                             // for unit testing only

    //  entity prefixes
    //
    //  Asop = asset operator
    //  Cm   = commodity
    //  Cx   = context
    //  Gate = gateway
    //  Teas = technical asset

    //  commodity abbreviations, with quantifying extensity
    //
    //  Cert = CmCarbonCert        kg
    //  Cseq = CmCarbonSeq         kg
    //  Elec = CmElectricity        J
    //  Fund = CmFunds              $ (generic unit of account)
    //  Heat = CmHeat               J
    //  Oxid = CmOxidize           kg
    //  Thrm = CmThermalFluid       J
    //  Work = CmWork               J
    //  Fiss = CmFission           kg (not generally supported)
    //  Land = CmProductiveLand    m2 (not generally supported)

    // test entities

    EFIRE("TestEntity0"                     , TestEntity0                     );
    EFIRE("TestEntity1"                     , TestEntity1                     );
    EFIRE("TestEntity2"                     , TestEntity2                     );

    // high-level entities

    EFIRE("TimeHorizon"                     , TimeHorizon                     );
    EFIRE("Overseer"                        , Overseer                        );
    EFIRE("DomainController"                , DomainController                );

    // operators

    EFIRE("AsopBasic"                       , AsopBasic                       );
    EFIRE("AsopGrid"                        , AsopGrid                        );
    EFIRE("AsopInelasticTs"                 , AsopInelasticTs                 );
```

```
    EFIRE("AsopInternalCosts"               , AsopInternalCosts                 );
    EFIRE("AsopLmpBidDialog"                , AsopLmpBidDialog                  );
    EFIRE("AsopLmpBidStatedTs1"             , AsopLmpBidStatedTs1               );
    EFIRE("AsopPrescribedOrder"             , AsopPrescribedOrder               );
    EFIRE("AsopLmpBidAdaptive1"             , AsopLmpBidAdaptive1               );
    EFIRE("AsopAdaptiveTs"                  , AsopAdaptiveTs                    );

    // commodities

    EFIRE("CmCarbonCert"                    , CmCarbonCert                      );
    EFIRE("CmCarbonSeq"                     , CmCarbonSeq                       );
    EFIRE("CmElectricity"                   , CmElectricity                     );
    EFIRE("CmFunds"                         , CmFunds                           );
    EFIRE("CmHeat"                          , CmHeat                            );
    EFIRE("CmOxidize"                       , CmOxidize                         );
    EFIRE("CmThermalFluid"                  , CmThermalFluid                    );
    EFIRE("CmWork"                          , CmWork                            );
    EFIRE("CmFission"                       , CmFission                         );
    EFIRE("CmProductiveLand"                , CmProductiveLand                  );

    EFIRE("CmOxidBiocoal"                   , CmOxidBiocoal                     );

    // gates

    EFIRE("GateStatedTariff:Cert"           , GateStatedTariff<CmCarbonCert>    );
    EFIRE("GateStatedTariff:Cseq"           , GateStatedTariff<CmCarbonSeq>     );
    EFIRE("GateStatedTariff:Elec"           , GateStatedTariff<CmElectricity>   );
    EFIRE("GateStatedTariff:Fund"           , GateStatedTariff<CmFunds>         );
    EFIRE("GateStatedTariff:Heat"           , GateStatedTariff<CmHeat>          );
    EFIRE("GateStatedTariff:Oxid"           , GateStatedTariff<CmOxidize>       );
    EFIRE("GateStatedTariff:Thrm"           , GateStatedTariff<CmThermalFluid>  );
    EFIRE("GateStatedTariff:Work"           , GateStatedTariff<CmWork>          );

    // technical assets

    EFIRE("TeasSource:Cert"                 , TeasSource<CmCarbonCert>          );
    EFIRE("TeasSource:Cseq"                 , TeasSource<CmCarbonSeq>           );
    EFIRE("TeasSource:Elec"                 , TeasSource<CmElectricity>         );
    EFIRE("TeasSource:Fund"                 , TeasSource<CmFunds>               );
    EFIRE("TeasSource:Heat"                 , TeasSource<CmHeat>                );
    EFIRE("TeasSource:Oxid"                 , TeasSource<CmOxidize>             );
    EFIRE("TeasSource:Thrm"                 , TeasSource<CmThermalFluid>        );
    EFIRE("TeasSource:Work"                 , TeasSource<CmWork>                );

    EFIRE("TeasLoad:Cert"                   , TeasLoad<CmCarbonCert>            );
    EFIRE("TeasLoad:Cseq"                   , TeasLoad<CmCarbonSeq>             );
    EFIRE("TeasLoad:Elec"                   , TeasLoad<CmElectricity>           );
    EFIRE("TeasLoad:Fund"                   , TeasLoad<CmFunds>                 );
    EFIRE("TeasLoad:Heat"                   , TeasLoad<CmHeat>                  );
    EFIRE("TeasLoad:Oxid"                   , TeasLoad<CmOxidize>               );
    EFIRE("TeasLoad:Thrm"                   , TeasLoad<CmThermalFluid>          );
    EFIRE("TeasLoad:Work"                   , TeasLoad<CmWork>                  );

    EFIRE("TeasHvTransmission"              , TeasHvTransmission                );
    EFIRE("TeasLoadElecTs"                  , TeasLoadElecTs                    );
    EFIRE("TeasMineElec"                    , TeasMineElec                      );
    EFIRE("TeasMineOxid"                    , TeasMineOxid                      );
    EFIRE("TeasOxidToElec"                  , TeasOxidToElec                    );
    EFIRE("TeasWindfarm"                    , TeasWindfarm                      );

    // nodes

    EFIRE("Node1Inj:Cert"                   , Node1Inj<CmCarbonCert>            );
    EFIRE("Node1Inj:Cseq"                   , Node1Inj<CmCarbonSeq>             );
    EFIRE("Node1Inj:Elec"                   , Node1Inj<CmElectricity>           );
    EFIRE("Node1Inj:Fund"                   , Node1Inj<CmFunds>                 );
    EFIRE("Node1Inj:Heat"                   , Node1Inj<CmHeat>                  );
    EFIRE("Node1Inj:Oxid"                   , Node1Inj<CmOxidize>               );
    EFIRE("Node1Inj:Thrm"                   , Node1Inj<CmThermalFluid>          );
    EFIRE("Node1Inj:Work"                   , Node1Inj<CmWork>                  );

    EFIRE("Node1Xit:Cert"                   , Node1Xit<CmCarbonCert>            );
    EFIRE("Node1Xit:Cseq"                   , Node1Xit<CmCarbonSeq>             );
    EFIRE("Node1Xit:Elec"                   , Node1Xit<CmElectricity>           );
    EFIRE("Node1Xit:Fund"                   , Node1Xit<CmFunds>                 );
    EFIRE("Node1Xit:Heat"                   , Node1Xit<CmHeat>                  );
    EFIRE("Node1Xit:Oxid"                   , Node1Xit<CmOxidize>               );
    EFIRE("Node1Xit:Thrm"                   , Node1Xit<CmThermalFluid>          );
    EFIRE("Node1Xit:Work"                   , Node1Xit<CmWork>                  );

    EFIRE("Node2Nul:Cert"                   , Node2Nul<CmCarbonCert>            );
```

```
    EFIRE("Node2Nul:Cseq"                   , Node2Nul<CmCarbonSeq>             );
    EFIRE("Node2Nul:Elec"                   , Node2Nul<CmElectricity>           );
    EFIRE("Node2Nul:Fund"                   , Node2Nul<CmFunds>                 );
    EFIRE("Node2Nul:Heat"                   , Node2Nul<CmHeat>                  );
    EFIRE("Node2Nul:Oxid"                   , Node2Nul<CmOxidize>               );
    EFIRE("Node2Nul:Thrm"                   , Node2Nul<CmThermalFluid>          );
    EFIRE("Node2Nul:Work"                   , Node2Nul<CmWork>                  );

    EFIRE("Node2Inj:Cert"                   , Node2Inj<CmCarbonCert>            );
    EFIRE("Node2Inj:Cseq"                   , Node2Inj<CmCarbonSeq>             );
    EFIRE("Node2Inj:Elec"                   , Node2Inj<CmElectricity>           );
    EFIRE("Node2Inj:Fund"                   , Node2Inj<CmFunds>                 );
    EFIRE("Node2Inj:Heat"                   , Node2Inj<CmHeat>                  );
    EFIRE("Node2Inj:Oxid"                   , Node2Inj<CmOxidize>               );
    EFIRE("Node2Inj:Thrm"                   , Node2Inj<CmThermalFluid>          );
    EFIRE("Node2Inj:Work"                   , Node2Inj<CmWork>                  );

    EFIRE("Node2Xit:Cert"                   , Node2Xit<CmCarbonCert>            );
    EFIRE("Node2Xit:Cseq"                   , Node2Xit<CmCarbonSeq>             );
    EFIRE("Node2Xit:Elec"                   , Node2Xit<CmElectricity>           );
    EFIRE("Node2Xit:Fund"                   , Node2Xit<CmFunds>                 );
    EFIRE("Node2Xit:Heat"                   , Node2Xit<CmHeat>                  );
    EFIRE("Node2Xit:Oxid"                   , Node2Xit<CmOxidize>               );
    EFIRE("Node2Xit:Thrm"                   , Node2Xit<CmThermalFluid>          );
    EFIRE("Node2Xit:Work"                   , Node2Xit<CmWork>                  );

    // junctions

    EFIRE("JuncDemand2Split:Cert"           , JuncDemand2Split<CmCarbonCert>    );
    EFIRE("JuncDemand2Split:Cseq"           , JuncDemand2Split<CmCarbonSeq>     );
    EFIRE("JuncDemand2Split:Elec"           , JuncDemand2Split<CmElectricity>   );
    EFIRE("JuncDemand2Split:Fund"           , JuncDemand2Split<CmFunds>         );
    EFIRE("JuncDemand2Split:Heat"           , JuncDemand2Split<CmHeat>          );
    EFIRE("JuncDemand2Split:Oxid"           , JuncDemand2Split<CmOxidize>       );
    EFIRE("JuncDemand2Split:Thrm"           , JuncDemand2Split<CmThermalFluid>  );
    EFIRE("JuncDemand2Split:Work"           , JuncDemand2Split<CmWork>          );

    EFIRE("JuncDemand2Join:Cert"            , JuncDemand2Join<CmCarbonCert>     );
    EFIRE("JuncDemand2Join:Cseq"            , JuncDemand2Join<CmCarbonSeq>      );
    EFIRE("JuncDemand2Join:Elec"            , JuncDemand2Join<CmElectricity>    );
    EFIRE("JuncDemand2Join:Fund"            , JuncDemand2Join<CmFunds>          );
    EFIRE("JuncDemand2Join:Heat"            , JuncDemand2Join<CmHeat>           );
    EFIRE("JuncDemand2Join:Oxid"            , JuncDemand2Join<CmOxidize>        );
    EFIRE("JuncDemand2Join:Thrm"            , JuncDemand2Join<CmThermalFluid>   );
    EFIRE("JuncDemand2Join:Work"            , JuncDemand2Join<CmWork>           );

    // contexts

    EFIRE("CxAmbientAirSim"                 , CxAmbientAirSim                   );
    EFIRE("CxAmbientAirTs"                  , CxAmbientAirTs                    );
    EFIRE("CxAmbientSolarTs"                , CxAmbientSolarTs                  );
    EFIRE("CxCommercialFix"                 , CxCommercialFix                   );

#endif // _XUTEST

#undef EFIRE                                // just the identifier is sufficient

// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  } // function 'registerEntyCreators'

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : asset.cc
//  file-create-date : Tue 26-Aug-2008 14:15 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : technical asset entity / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "teas.h"             // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../b/bandtaf.h"     // banded tariff set and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  MEMBER FUNCTION : TechnicalAsset
// ----------------------------------------------------------

TechnicalAsset::TechnicalAsset
(const std::string entityId,
 Record&           record,
 const int         commitmentModeSum) :
  CostRegister(record),
  Block(entityId, record),
  TicToc(commitmentModeSum),
  d_ceilingDuty(-1.0),                      // nonsensical value
  d_floorDuty(-1.0),                        // nonsensical value
  d_cogenHeatLeadWeight(0.0)                // zero is power led
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~TechnicalAsset
// ----------------------------------------------------------

TechnicalAsset::~TechnicalAsset()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
```

```
//  MEMBER FUNCTION : getCeilingDuty
// -------------------------------------------------------
//  Description  : get current ceiling duty (conceptually equivalent to current capacity)
//  Role         : called by asset operators, most likely those preparing bid sets
//  Techniques   : (nothing special)
//  Status       : complete
//
//  Design notes
//
//      If required, 'd_ceilingDuty' should be set in the
//      relevant technical asset constrain call.
//
//      It is most likely that asset operators requiring this
//      information will be preparing LMP (nodal pricing) bid
//      sets.
//
// -------------------------------------------------------

double
TechnicalAsset::getCeilingDuty() const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", d_ceilingDuty);

  // integrity checks
  if ( d_ceilingDuty < 0 )
    {
      s_logger->repx(logga::warn, "ceiling duty not reset", getIdAndKind());
    }

  // return
  return d_ceilingDuty;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getFloorDuty
// -------------------------------------------------------
//
//  see documentation for 'getCeilingDuty'
//
// -------------------------------------------------------

double
TechnicalAsset::getFloorDuty() const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", d_floorDuty);

  // integrity checks
  if ( d_floorDuty < 0 )
    {
      s_logger->repx(logga::warn, "floor duty not reset", getIdAndKind());
    }

  // return
  return d_floorDuty;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : setCogenHeatWeight
// -------------------------------------------------------

void
TechnicalAsset::setCogenHeatWeight
(const double cogenHeatLeadWeight)
{
  s_logger->repx(logga::adhc, "entering member function, weighting", cogenHeatLeadWeight);
  d_cogenHeatLeadWeight = cogenHeatLeadWeight;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getCogenHeatWeight
// -------------------------------------------------------

const double
TechnicalAsset::getCogenHeatWeight() const
{
  return d_cogenHeatLeadWeight;
}

// -------------------------------------------------------
```

```
//  MEMBER FUNCTION : getPriorDuty
// -------------------------------------------------------

double                                          // prior duty
TechnicalAsset::getPriorDuty() const            // 'NaN' for step 0
{
  return d_dutyStats.last();                    // 'NaN' if object not yet filled
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getPriorSize
// -------------------------------------------------------

double                                          // prior size
TechnicalAsset::getPriorSize() const            // 'NaN' for step 0
{
  return d_sizeStats.last();                    // 'NaN' if object not yet filled
}

// -------------------------------------------------------
//  MEMBER FUNCTION : obtainTariffSet
// -------------------------------------------------------

shared_ptr<BandedTariffSet>                 // current tariff set for adaptive behavior
TechnicalAsset::obtainTariffSet() const
{
  // warning
  s_logger->repx(logga::warn, "function should have been redefined", "");

  // active code
  return shared_ptr<BandedTariffSet>();
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas1.cc
//  file-create-date : Wed 15-Apr-2009 21:03 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "teas01.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optops.h"      // operate optimization sub-problems for hard assets
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasHvTransmission
// ----------------------------------------------------------

TeasHvTransmission::TeasHvTransmission
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  CostRegisterEmbFin(record),
  d_injectCapacity(record.tieSingle<double>("inject-capacity")),
  d_voltage(record.tieSingle<double>("voltage")),
  d_ohmsPerMetre(record.tieSingle<double>("ohms-per-meter")),
  d_length(record.tieSingle<double>("length")),
  d_discretizationSteps(record.tieSingle<int>("discretization-steps")),
  d_gridCommodity(record.tieSingle<std::string>("grid-commodity")),
  d_directions(record.tieTimeseries<bool>("directions")),
  d_injections(record.tieTimeseries<double>("injections")),
  d_exits(record.tieTimeseries<double>("exits")),
  d_relativeDutys(record.tieTimeseries<double>("relative-dutys")),
  d_relativeLosss(record.tieTimeseries<double>("relative-losss")),
  d_cable(Cable<CmElectricity>::create
```

```
          (entityId,                          // me
           record.tieSingle<std::string>("socket-1"),
           d_gridCommodity)),                 // common value
  d_socket(Socket<CmElectricity>::create
           (entityId,                         // me
            "elec-1",                         // hardcoded socket label
            d_gridCommodity)),                // common value
  d_ops()                                     // empty pointer
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ~TeasHvTransmission
// ----------------------------------------------------------

TeasHvTransmission::~TeasHvTransmission()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : technical asset constrain call
//  Role         : characterize current engineering and short-run costs
//  Techniques   : (nothing special)
//  Status       : first-pass complete
// ----------------------------------------------------------

const int                                     // return zero
TeasHvTransmission::constrain
(const xeona::DomainMode capacityMode)
{
  // additional initial reporting as appropriate
  // YEEK 30 CODE (set by '--yeek')
  if ( xeona::yeek == 30 || xeona::yeek == 1 )
    {
      const std::string subtype = xeona::demangle(typeid(*this).name()); // CAUTION: deref
      s_logger->repx(logga::adhc, "entering member function", subtype);
    }
  else
    {
      s_logger->repx(logga::adhc, "entering member function", "TeasHvTransmission");
    }

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsTransmission(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define some global cols for internal use
  int cabGol = -1;
  int socGol = -1;

  // upload the engineering -- in this case, the directionality
  // is defined by my cable-to-socket arrangement
  boost::tie(cabGol,
             socGol)
    = d_ops->uploadEngineering(d_injectCapacity,
                               d_voltage,
                               d_ohmsPerMetre,
                               d_length,
                               d_discretizationSteps);

  // upload specific costs -- including increment the "shift"
  // term note also the 50/50 split on costs -- see r4150 and
  // back for previous call chain
  d_ops->uploadShortrunCosts(d_dutySpecCosts * 0.5, cabGol);
  d_ops->uploadShortrunCosts(d_dutySpecCosts * 0.5, socGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // bind global cols to the relevant interfaces
  d_cable ->bindOsp(d_solver, cabGol);
  d_socket->bindOsp(d_solver, socGol);

  // store duty values
```

```
  d_ceilingDuty = +d_injectCapacity;         // automatically decoupled from 'd_capacity'
  d_floorDuty   = -d_injectCapacity;

  // additional reporting as appropriate
  // YEEK 13 CODE (set by '--yeek')
  if ( xeona::yeek == 13 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  transmission constrain (note scaling)"                          << "\n"
          << std::fixed << std::setprecision(2)
          << "    inject capacity [MW]     : " << ( d_injectCapacity / 1.0e+06 ) << "\n"
          << "    voltage [kV]             : " << ( d_voltage        / 1.0e+03 ) << "\n"
          << "    resistance [micro-ohm/m] : " << ( d_ohmsPerMetre   / 1.0e-06 ) << "\n"
          << "    length [km]              : " << ( d_length         / 1.0e+03 ) << "\n"
          << "    discretization steps [-] : " << d_discretizationSteps          << "\n"
          << std::fixed << std::setprecision(2) << std::showpos
          << "    ceiling duty (unscaled)  : " << d_ceilingDuty                  << "\n"
          << "    floor   duty (unscaled)  : " << d_floorDuty                    << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // set duty to zero as not needed in this case, also capture return value
  const int dutyGol = 0;                     // zero here means no need to couple

  // return zero
  return dutyGol;

} // function 'TeasHvTransmission::constrain'

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------

void
TeasHvTransmission::washup()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", d_step);

  // results recovery
  bool   direction    = false;
  double flowCable    =  0.0;                // negative values are valid
  double flowSocket   =  0.0;                // negative values are valid
  double relativeDuty = -1.0;                // nonsensical value
  double relativeLoss = -1.0;                // nonsensical value

  boost::tie(direction,                      // flow direction flag, 'true' = forward
             flowCable,                      // cable flow [W], can be negative
             flowSocket,                     // socket flow [W]
             relativeDuty,                   // relative duty [-]
             relativeLoss)                   // relative size [-]
    = d_ops->downloadSolution();

  // store entity state information (code needs confirmation)
  d_directions->at(d_step) = direction;
  if ( direction == true )
    {
      d_injections->at(d_step) = +flowCable;
      d_exits     ->at(d_step) = +flowSocket;
    }
  else
    {
      d_injections->at(d_step) = -flowSocket;
      d_exits     ->at(d_step) = -flowCable;
    }
  d_relativeDutys->at(d_step)  = relativeDuty;
  d_relativeLosss->at(d_step)  = relativeLoss;

  double duty = std::max(std::abs(flowCable), std::abs(flowSocket));

  // additional reporting as appropriate
  // YEEK 13 CODE (set by '--yeek')
  if ( xeona::yeek == 13 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << "  transmission solution"                                  << "\n"
          << "    direction {0,1} : " << direction                      << "\n"
          << std::fixed << std::setprecision(0) << std::showpos
          << "    cable flow      : " << flowCable                      << "\n"
          << "    socket flow     : " << flowSocket                     << "\n"
```

```
          << std::fixed << std::setprecision(5) << std::noshowpos
          << "    relative duty   : " << relativeDuty                << "\n"
          << "    relative loss   : " << relativeLoss                << "\n"
          << std::fixed << std::setprecision(0) << std::showpos
          << "    injection       : " << d_injections->at(d_step)    << "\n"
          << "    exit            : " << d_exits->at(d_step)         << "\n"
          << "    duty (abs'ed)   : " << duty                        << "\n";
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->putx(logga::dbug, put);
    }

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts                         // polymorphic 'CostRegister' call
    (d_step,                                  // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // embedded costs processing, outer call also updates instance data
  updateEmbeddedCosts                         // polymorphic 'CostRegister' call
    (d_step);                                 // provided by class 'TicToc'

  // store some on-the-fly statistics
  d_dutyStats(duty);                          // functor provided by class 'Block'
  d_sizeStats(d_injectCapacity);              // functor provided by class 'Block'
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas02.cc
//  file-create-date : Wed 22-Apr-2009 12:40 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 2 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas02.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "teas02.h"           // companion header for this file (place first)

#include "../e/cxamb01.h"     // concrete ambient conditions contexts 1
#include "../c/tsops.h"       // overloaded operators and similar for timeseries
#include "../c/stats.h"       // on-the-fly statistical calculations
#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optops.h"      // operate optimization sub-problems for hard assets
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <typeinfo>           // run-time type information (RTTI)

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasOxidToElec
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasOxidToElec
// ----------------------------------------------------------

TeasOxidToElec::TeasOxidToElec
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  CostRegisterEmbFin(record),
  d_prodLoBound(record.tieSingle<double>("prod-lo-bound")),
  d_prodHiBound(record.tieSingle<double>("prod-hi-bound")),
  d_marginalEfficiency(record.tieSingle<double>("marginal-efficiency")),
```

```
  d_fuelNoload(record.tieSingle<double>("fuel-noload")),
  d_fuelAncillary(record.tieSingle<double>("fuel-ancillary")),
  d_rampRestraintDown(record.tieSingle<double>("ramp-restraint-down")),
  d_rampRestraintUp(record.tieSingle<double>("ramp-restraint-up")),
  d_productions(record.tieTimeseries<double>("productions")),
  d_shutdownStatuss(record.tieTimeseries<bool>("shutdown-statuss")),
  d_inOxid(Cable<CmOxidize>::create
           (entityId,                          // me
            record.tieSingle<std::string>("socket-oxidize"),
            record.tieSingle<std::string>("cable-oxidize-commodity"))),
  d_outElec(Socket<CmElectricity>::create
            (entityId,                         // me
             "elec-1",                         // hardcoded socket label
             record.tieSingle<std::string>("socket-electricity-commodity"))),
  d_oxid(),
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";

  // integrity checks
  if ( d_rampRestraintDown < 0.0 )
    {
      s_logger->repx(logga::warn,
                     "negative downward ramp restraint",
                     d_rampRestraintDown);
    }
  if ( d_rampRestraintUp < 0.0 )
    {
      s_logger->repx(logga::warn,
                     "negative upward ramp restraint",
                     d_rampRestraintUp);
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~TeasOxidToElec
// ----------------------------------------------------------

TeasOxidToElec::~TeasOxidToElec()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : establish
// ----------------------------------------------------------

void
TeasOxidToElec::establish()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // CAUTION: 'Interface::getCm' can only be called AFTER the
  // interface connections are complete -- which is why this code
  // is here and not in the constructor

  // obtain fuel commodity
  const shared_ptr<Commodity> com = d_inOxid->getCm();
  d_oxid                          = dynamic_pointer_cast<CmOxidize>(com);

  // extra reporting during development
  s_logger->repx(logga::adhc,
                 "combustion enthalpy recovered [J/kg]",
                 d_oxid->getSpecCombEnthalpy());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : technical asset constrain call
//  Role         : characterize current engineering and short-run costs
//  Techniques   : 'boost::tuples::ignore' to ignore tuple ties
//  Status       : first-pass complete
// ----------------------------------------------------------

const int                                  // duty gol
```

```
TeasOxidToElec::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "TeasOxidToElec");

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // obtain fuel attributes
  const double specEnthalpy = d_oxid->getSpecCombEnthalpy();     // about 20e+06 J/kg
  const double specCo2equiv = d_oxid->getSpecCo2equiv();         // for carbon, 3.7 kg/kg

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac1Out1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define global cols for internal use
  int  inGol = -1;                          // nonsensical value
  int outGol = -1;                          // nonsensical value

  // calculate ramp rate restrictions
  double rampLoSize = -1.0;                 // nonsensical value
  double rampHiSize = -1.0;                 // nonsensical value
  if ( d_step == 0 )                        // no prior information
    {
      rampLoSize = 0.0;
      rampHiSize = d_prodHiBound;
    }
  else                                      // modify based on previous production
    {
      const double priorProduction = d_productions->at(d_step - 1);
      rampLoSize = priorProduction - (d_nameplateSize * d_rampRestraintDown);
      rampHiSize = priorProduction + (d_nameplateSize * d_rampRestraintUp);
    }

  // upload the engineering
  boost::tie(inGol,                         // factor (fuel) stream
             outGol,                        // output (product) stream
             boost::tuples::ignore)         // trip status variable
    = d_ops->uploadEngineering(d_prodLoBound,
                               d_prodHiBound,
                               d_marginalEfficiency,
                               d_fuelNoload,
                               d_fuelAncillary,
                               rampLoSize,
                               rampHiSize,
                               specEnthalpy); // kg->J conversion

  // upload specific costs -- including increment the "shift" term
  d_dutySpecCosts.ghg = specCo2equiv;
  d_ops->uploadShortrunCosts(d_dutySpecCosts, inGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // ad-hoc reporting
  s_logger->repx(logga::adhc, "duty-specific ghg cost", d_dutySpecCosts.ghg);

  // bind global cols to the relevant interfaces
  d_inOxid ->bindOsp(d_solver,  inGol);
  d_outElec->bindOsp(d_solver, outGol);

  // store duty values
  d_floorDuty   = d_prodLoBound;
  d_ceilingDuty = d_prodHiBound;

  // return the duty gol
  return outGol;
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : washup
// -----------------------------------------------------------
//  Description  : technical asset washup call
//  Role         : recover and record solution
//  Techniques   : (nothing special)
//  Status       : first-pass complete
// -----------------------------------------------------------

void
TeasOxidToElec::washup()
```

```
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", d_step);

  // results recovery
  double fuelUsage        = -1.0;            // nonsensical value
  double actualProduction = -1.0;            // nonsensical value
  bool   tripStatus       = false;           // arbitrary value
  boost::tie(fuelUsage,
             actualProduction,
             tripStatus) = d_ops->downloadSolution();

  // store entity state information
  d_productions->at(d_step)     = actualProduction;
  d_shutdownStatuss->at(d_step) = tripStatus;

  // another local variable
  const double potentialProduction = d_prodHiBound;

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts                        // polymorphic 'CostRegister' call
    (d_step,                                 // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // ad-hoc reporting
  const CostSet var = d_ops->downloadShortrunCosts().get<0>();
  s_logger->repx(logga::adhc, "variable ghg cost", var.ghg);

  // embedded costs processing, outer call also updates instance data
  updateEmbeddedCosts                        // polymorphic 'CostRegister' call
    (d_step);                                // provided by class 'TicToc'

  // store some on-the-fly statistics
  d_dutyStats(actualProduction);             // functor provided by class 'Block'
  d_sizeStats(potentialProduction);          // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  CLASS           : TeasWindfarm
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasWindfarm
// ----------------------------------------------------------

TeasWindfarm::TeasWindfarm
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  CostRegisterEmbFin(record),
  d_ambientAirContext(record.lazyLink<CxAmbientAir>("ambient-air-context")),
  d_count(record.tieSingle<int>("count")),
  d_turbineRating(record.tieSingle<double>("turbine-rating")),
  d_loCutSpeed(record.tieSingle<double>("lo-cut-speed")),
  d_hiCutSpeed(record.tieSingle<double>("hi-cut-speed")),
  d_potentialProductions(record.tieTimeseries<double>("potential-productions")),
  d_actualProductions(record.tieTimeseries<double>("actual-productions")),
  d_availability(record.tieSingle<double>("availability")),
  d_spill(record.tieSingle<double>("spill")),
  d_outElec(Socket<CmElectricity>::create
            (entityId,                       // me
             "elec-1",                       // hardcoded socket label
             record.tieSingle<std::string>("socket-electricity-commodity"))),
  d_ops()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~TeasWindfarm
// ----------------------------------------------------------

TeasWindfarm::~TeasWindfarm()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
```

```
//  MEMBER FUNCTION : establish
// -------------------------------------------------------
//  Description  : low priority development reporting
//  Role         : logging output solely
//  Techniques   :
//  Status       : complete
// -------------------------------------------------------

void
TeasWindfarm::establish()
{
  // CAUTION: this code provides only low priority development
  // reporting -- it should not be used as a template for other
  // 'establish' calls

  // describe  ambient air context
  std::ostringstream put;
  s_logger->repx(logga::adhc, "development reporting 1 follows", "");
  put << "  post-link identifier : " << d_ambientAirContext->getIdentifier() << "\n"
      << "  post-link resource   : " << d_ambientAirContext.get()            << "\n";
  s_logger->putx(logga::adhc, put);

  // get and report the current windspeed
  // CAUTION: cannot use 'd_step' as it stills remains at its -1 initiation
  const int index   = 0;
  const double wind = d_ambientAirContext->getWindSpeed(index);

  s_logger->repx(logga::adhc, "development reporting 2 follows", "");
  put << "  link test : 'CxAmbientAir' linked entity from 'TeasWindfarm' entity" << "\n"
      << "    fixed index (step not yet set) : " << index                       << "\n"
      << "    wind-speed                     : " << wind                        << "\n";
  s_logger->putx(logga::adhc, put);
}

// -------------------------------------------------------
//  MEMBER FUNCTION : constrain
// -------------------------------------------------------
//  Description  : constrain call
//  Role         : characterize current engineering and short-run costs
//  Techniques   : protected member function 'calcTurbinePower'
//  Status       : complete
// -------------------------------------------------------

const int                                  // duty gol
TeasWindfarm::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "TeasWindfarm");

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac0Out1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define a global col for internal use
  int outGol = -1;                         // nonsensical value

  // get prevailing wind-speed
  const double wspeed  = d_ambientAirContext->getWindSpeed(d_step);

  // calculate power range
  const double loPower = 0.0;
  const double hiPower = d_count * calcTurbinePower(wspeed);
  s_logger->repx(logga::xtra, "turbine count", d_count);
  s_logger->repx(logga::xtra, "potential park production, hiPower", hiPower);

  // upload the engineering
  boost::tie(outGol) = d_ops->uploadEngineering(loPower, hiPower);

  // upload specific costs -- including increment the "shift" term
  d_ops->uploadShortrunCosts(d_dutySpecCosts, outGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // bind global cols to the relevant interfaces
  d_outElec->bindOsp(d_solver, outGol);

  // store values
```

```
  d_potentialProductions->at(d_step) = hiPower;

  // store duty values
  d_floorDuty   = loPower;
  d_ceilingDuty = hiPower;

  // return the duty gol
  return outGol;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------
//  Description  : technical asset washup call
//  Role         : recover and record solution
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

void
TeasWindfarm::washup()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", d_step);

  // results recovery
  double actualProduction      = -1.0;       // nonsensical value
  boost::tie(actualProduction) = d_ops->downloadSolution();

  // store entity state information
  d_actualProductions->at(d_step) = actualProduction;

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts                        // polymorphic 'CostRegister' call
    (d_step,                                 // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // embedded costs processing, outer call also updates instance data
  updateEmbeddedCosts                        // polymorphic 'CostRegister' call
    (d_step);                                // provided by class 'TicToc'

  // store some on-the-fly statistics
  const double potentialProduction = d_potentialProductions->at(d_step);
  d_dutyStats(actualProduction);             // functor provided by class 'Block'
  d_sizeStats(potentialProduction);          // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : conclude
// ----------------------------------------------------------

void
TeasWindfarm::conclude()
{
  // availability = potential production / nameplate production (spill not considered)
  const Statistics<double> potproStats = xeona::fillStatistics(d_potentialProductions);
  d_availability                       = potproStats.mean() / (d_count * d_turbineRating);

  // normalized spill = discarded production / potential production
  const double totalPotential = xeona::vectorSum(d_potentialProductions);
  const double totalActual    = xeona::vectorSum(d_actualProductions);
  d_spill                     = (totalPotential - totalActual) / totalPotential;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : calcTurbinePower
// ----------------------------------------------------------
//  Description  : returns power for single turbine
//  Role         : utility function
//  Techniques   : maths
//  Status       : complete
//
//  Formula
//
//      cube-law mapping windspeed 's' to power 'p':
//
//          p = A * s^3 + B when lo < s < hi, else p = 0
//
//          with constants A = C/hi^3 and B = 0
//
//      where 'C' is the turbine rating, 'lo' is the low
```

```
//      cut-in/out and 'hi' is the high cut-in/out (hysteresis
//      not modeled)
//
//      the cube-law is from physics and runs thru the origin
//      (that is, zero power at zero speed)
//
// -------------------------------------------------------

double                                      // resultant power [W]
TeasWindfarm::calcTurbinePower              // single turbine
(const double windSpeed)                    // known wind speed [m/s]
{
  double power = 0.0;                       // zero output
  if ( windSpeed > d_loCutSpeed && windSpeed < d_hiCutSpeed )
    {
      // turbine is within its working range
      const double normSpeed = windSpeed / d_hiCutSpeed;
      power                  = d_turbineRating * std::pow(normSpeed, 3);
      s_logger->repx(logga::xtra, "turbine within working range", power);
    }
  else
    {
      std::ostringstream oss;
      oss << d_loCutSpeed << " - " << d_hiCutSpeed;
      s_logger->repx(logga::xtra, "turbine outside working range", oss.str());
    }
  return power;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas03.cc
//  file-create-date : Sat 16-May-2009 12:24 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 3 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas03.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "teas03.h"           // companion header for this file (place first)

#include "../c/util3.h"       // free functions for floating point comparison
#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optops.h"      // operate optimization sub-problems for hard assets

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasLoadElecTs
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasLoadElecTs
// ----------------------------------------------------------

TeasLoadElecTs::TeasLoadElecTs
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  CostRegisterEmbFin(record),
  d_loads(record.tieTimeseries<double>("loads")),
  d_inElec(Cable<CmElectricity>::create
           (entityId,                          // me
            record.tieSingle<std::string>("socket-electricity"),
            record.tieSingle<std::string>("cable-electricity-commodity"))),
  d_ops()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
```

```
  d_builtinRemark = "beta / cost output inactive / superseded, try TeasLoad:Elec";
  s_logger->repx(logga::rankJumpy,
                 "superseded entity, try",
                 "TeasLoad:Elec + AsopInelasticTs");
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ~TeasLoadElecTs
// ----------------------------------------------------------

TeasLoadElecTs::~TeasLoadElecTs()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : incomplete
// ----------------------------------------------------------

const int                                       // duty gol
TeasLoadElecTs::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "TeasLoadElecTs");

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac1Out0(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define a global col for internal use
  int inGol = -1;                               // nonsensical value

  // get prevailing load
  const double load = d_loads->at(d_step);
  s_logger->repx(logga::xtra, "current load", load);

  // upload the engineering (using single argument inflexible call)
  boost::tie(inGol) = d_ops->uploadEngineering(load);

  // upload specific costs -- including increment the "shift" term
  d_ops->uploadShortrunCosts(d_dutySpecCosts, inGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // bind global cols to the relevant interfaces
  d_inElec->bindOsp(d_solver, inGol);

  // store duty values
  d_floorDuty   = load;
  d_ceilingDuty = load;

  // return the duty gol
  return inGol;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------
//  Description  : washup call
//  Role         : obtain and process results
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

void
TeasLoadElecTs::washup()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", d_step);

  // grab requested load
  const double currentLoad      = d_loads->at(d_step);
```

```
  // results recovery
  double actualConsumption      = -1.0;      // nonsensical value
  boost::tie(actualConsumption) = d_ops->downloadSolution();

  // preform an integrity check using bounded equality
  if ( ! xeona::almostEqual(actualConsumption, currentLoad, xeona::numic) )
    {
      std::ostringstream oss;
      oss << actualConsumption << " : " << currentLoad;
      s_logger->repx(logga::warn, "consumption : load mismatch", oss.str());
    }

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts
    (d_step,                                // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // store some on-the-fly statistics
  d_dutyStats(actualConsumption);           // functor provided by class 'Block'
  d_sizeStats(currentLoad);                 // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  CLASS           : TeasMineOxid
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasMineOxid
// ----------------------------------------------------------

TeasMineOxid::TeasMineOxid
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  d_extractLoBound(record.tieSingle<double>("extract-lo-bound")),
  d_extractHiBound(record.tieSingle<double>("extract-hi-bound")),
  d_extractions(record.tieTimeseries<double>("extractions")),
  d_outOxid(Socket<CmOxidize>::create
            (entityId,                      // me
             "oxid-1",                      // hardcoded socket label
             record.tieSingle<std::string>("socket-oxide-commodity"))),
  d_ops()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta / superseded, try TeasSource:Oxid";
  s_logger->repx(logga::rankJumpy,
                 "superseded entity, try",
                 "TeasSource:Oxid + AsopInelasticTs");
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ~TeasMineOxid
// ----------------------------------------------------------

TeasMineOxid::~TeasMineOxid()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

const int                                   // duty gol
TeasMineOxid::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "TeasMineOxid");

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);
```

```
  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac0Out1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define a global col for internal use
  int outGol = -1;                            // nonsensical value

  // upload the engineering
  boost::tie(outGol) = d_ops->uploadEngineering(d_extractLoBound,
                                                d_extractHiBound);

  // upload specific costs -- including increment the "shift" term
  d_ops->uploadShortrunCosts(d_dutySpecCosts, outGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // store duty values
  d_floorDuty   = d_extractLoBound;
  d_ceilingDuty = d_extractHiBound;

  // bind global cols to the relevant interfaces
  d_outOxid->bindOsp(d_solver, outGol);

  // return the duty gol
  return outGol;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : washup
// ------------------------------------------------------------
//  Description  : washup call
//  Role         : obtain and process results
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

void
TeasMineOxid::washup()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", d_step);

  // results recovery
  double actualExtraction      = -1.0;      // nonsensical value
  boost::tie(actualExtraction) = d_ops->downloadSolution();

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts
    (d_step,                                  // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // store some on-the-fly statistics
  d_dutyStats(actualExtraction);              // functor provided by class 'Block'
  d_sizeStats(d_extractHiBound);              // functor provided by class 'Block'
}

// ------------------------------------------------------------
//  CLASS           : TeasMineElec
// ------------------------------------------------------------

// ------------------------------------------------------------
//  MEMBER FUNCTION : TeasMineElec
// ------------------------------------------------------------

TeasMineElec::TeasMineElec
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  d_extractLoBound(record.tieSingle<double>("extract-lo-bound")),
  d_extractHiBound(record.tieSingle<double>("extract-hi-bound")),
  d_extractions(record.tieTimeseries<double>("extractions")),
  d_outElec(Socket<CmElectricity>::create
            (entityId,                        // me
             "elec-1",                        // hardcoded socket label
             record.tieSingle<std::string>("socket-electricity-commodity"))),
  d_ops()
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta / superseded, try TeasSource:Elec";
```

```
  s_logger->repx(logga::rankJumpy,
                 "superseded entity, try",
                 "TeasSource:Elec + AsopInelasticTs");
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ~TeasMineElec
// ----------------------------------------------------------

TeasMineElec::~TeasMineElec()
{
  s_logger->repx(logga::adhc, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

const int                                       // duty gol
TeasMineElec::constrain
(const xeona::DomainMode capacityMode)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "TeasMineElec");

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac0Out1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define a global col for internal use
  int outGol = -1;                              // nonsensical value

  // upload the engineering
  boost::tie(outGol) = d_ops->uploadEngineering(d_extractLoBound,
                                                d_extractHiBound);

  // upload specific costs -- including increment the "shift" term
  d_ops->uploadShortrunCosts(d_dutySpecCosts, outGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // store duty values
  d_floorDuty   = d_extractLoBound;
  d_ceilingDuty = d_extractHiBound;

  // bind global cols to the relevant interfaces
  d_outElec->bindOsp(d_solver, outGol);

  // return the duty gol
  return outGol;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------
//  Description  : washup call
//  Role         : obtain and process results
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

void
TeasMineElec::washup()
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function, step", d_step);

  // results recovery
  double actualExtraction      = -1.0;      // nonsensical value
  boost::tie(actualExtraction) = d_ops->downloadSolution();

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts
```

```
    (d_step,                                  // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // store some on-the-fly statistics
  d_dutyStats(actualExtraction);              // functor provided by class 'Block'
  d_sizeStats(d_extractHiBound);              // functor provided by class 'Block'
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teas04.cc
//  file-create-date : Thu 08-Oct-2009 12:58 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets 4 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teas04.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "teas04.h"           // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optops.h"      // operate optimization sub-problems for hard assets
#include "../b/gate.h"        // gateway entity
#include "../b/commods.h"     // commodities hierarchy
#include "../b/bandtaf.h"     // banded tariff set and support

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : TeasLoad <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasLoad<C>
// ----------------------------------------------------------

template <typename C>                       // 'C' for commodity
TeasLoad<C>::TeasLoad
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  CostRegisterEmbFin(record),
  d_commodity(record.tieSingle<std::string>("cable-commodity")),
  d_demandHiBound(record.tieSingle<double>("demand-hi-bound")),
  d_loads(record.tieTimeseries<double>("loads")),
  d_inCommodity(Cable<C>::create
                (entityId,                  // me
```

```
                record.tieSingle<std::string>("socket-1"),
                d_commodity)),              // common value
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~TeasLoad<C>
// ------------------------------------------------------------

template <typename C>
TeasLoad<C>::~TeasLoad()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ------------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : incomplete
//
//  Design notes
//
//      The factor equality constraint is set by my operator.
//
// ------------------------------------------------------------

template <typename C>
const int                                   // duty gol
TeasLoad<C>::constrain
(const xeona::DomainMode capacityMode)
{
  // additional initial reporting as appropriate
  // YEEK 30 CODE (set by '--yeek')
  if ( xeona::yeek == 30 || xeona::yeek == 1 )
    {
      const std::string subtype = xeona::demangle(typeid(*this).name()); // CAUTION: deref
      s_logger->repx(logga::adhc, "entering member function", subtype);
    }
  else
    {
      s_logger->repx(logga::adhc, "entering member function", "TeasLoad<C>");
    }

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac1Out0(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define a global col for internal use
  int inGol = -1;                           // nonsensical value

  // upload the engineering (using the double argument flexible call)
  const double demandLoBound = 0.0;
  boost::tie(inGol) = d_ops->uploadEngineering(  demandLoBound,
                                                 d_demandHiBound);

  // upload specific costs -- including increment the "shift" term
  d_ops->uploadShortrunCosts(d_dutySpecCosts, inGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // bind global cols to the relevant interfaces
  d_inCommodity->bindOsp(d_solver, inGol);

  // store duty values
  d_floorDuty   =   demandLoBound;
  d_ceilingDuty = d_demandHiBound;

  // return the duty gol
```

```
  return inGol;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : washup
// ----------------------------------------------------------
//  Description  : washup call
//  Role         : obtain and process results
//  Techniques   : (nothing special)
//  Status       : complete
// ----------------------------------------------------------

template <typename C>
void
TeasLoad<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double load      = -1.0;                   // nonsensical value
  boost::tie(load) = d_ops->downloadSolution();

  // store entity state information
  d_loads->at(d_step) = load;

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts
    (d_step,                                 // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // store some on-the-fly statistics
  d_dutyStats(load);                         // functor provided by class 'Block'
  d_sizeStats(d_demandHiBound);              // functor provided by class 'Block'
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : obtainGateway
// ----------------------------------------------------------

template <typename C>
shared_ptr<GateCom<C> >
TeasLoad<C>::obtainGateway() const
{
  // preliminary reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code
  shared_ptr<Entity> entity = d_inCommodity->getPartner();
  if ( entity == 0 )
    {
      s_logger->repx(logga::adhc, "gateway as Entity not obtained", "");
    }
  else
    {
      s_logger->repx(logga::adhc, "gateway as Entity obtained", "");
    }
  shared_ptr<GateCom<C> > gateway = dynamic_pointer_cast<GateCom<C> >(entity);
  if ( gateway == 0 )
    {
      s_logger->repx(logga::warn, "GateCom <> cast failed", "");
    }
  return gateway;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : obtainTariffSet
// ----------------------------------------------------------

template <typename C>
shared_ptr<BandedTariffSet>
TeasLoad<C>::obtainTariffSet() const
{
  // preliminary reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code
  shared_ptr<GateCom<C> >     gateway = obtainGateway();
  shared_ptr<BandedTariffSet> tariffs = gateway->obtainTariffSet();
  if ( tariffs == 0 )
    {
```

```
      s_logger->repx(logga::warn, "tariffs not obtained", tariffs);
    }
  else
    {
      s_logger->repx(logga::adhc, "tariffs obtained", "");
    }

  // return
  return tariffs;
}

// ----------------------------------------------------------
//  CLASS           : TeasSource <>
// ----------------------------------------------------------

// ----------------------------------------------------------
//  MEMBER FUNCTION : TeasSource<C>
// ----------------------------------------------------------

template <typename C>                          // 'C' for commodity
TeasSource<C>::TeasSource
(const std::string entityId,
 Record&           record) :
  CostRegister(record),
  TechnicalAsset(entityId, record, xeona::e_commitmentModes),
  CostRegisterSRFin(record),
  CostRegisterEmbFin(record),
  d_commodity(record.tieSingle<std::string>("socket-commodity")),
  d_extractLoBound(record.tieSingle<double>("extract-lo-bound")),
  d_extractHiBound(record.tieSingle<double>("extract-hi-bound")),
  d_extractions(record.tieTimeseries<double>("extractions")),
  d_outCommodity(Socket<C>::create
                 (entityId,                    // me
                  "sock-1",                    // hardcoded socket label
                  d_commodity)),               // common value
  d_ops()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());

  // builtin remark
  d_builtinRemark = "beta";
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~TeasSource<C>
// ----------------------------------------------------------

template <typename C>
TeasSource<C>::~TeasSource()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : constrain
// ----------------------------------------------------------
//  Description  : engineering characterization
//  Role         : host call
//  Techniques   : (nothing special)
//  Status       : incomplete
//
//  Design notes
//
//      The factor equality constraint is set by my operator.
//
// ----------------------------------------------------------

template <typename C>
const int                                      // duty gol
TeasSource<C>::constrain
(const xeona::DomainMode capacityMode)
{
  // additional initial reporting as appropriate
  // YEEK 30 CODE (set by '--yeek')
  if ( xeona::yeek == 30 || xeona::yeek == 1 )
    {
      const std::string subtype = xeona::demangle(typeid(*this).name()); // CAUTION: deref
      s_logger->repx(logga::adhc, "entering member function", subtype);
    }
```

```
  else
    {
      s_logger->repx(logga::adhc, "entering member function", "TeasSource<C>");
    }

  // create and fill a label object
  const std::string teasId = getIdentifier();
  Label lab(teasId);

  // recreate new operations OSP of the required type
  d_ops.reset(new OpsFac0Out1(d_solver, d_commitmentMode));
  d_ops->loadOspLabel(lab.str());

  // define a global col for internal use
  int outGol = -1;                              // nonsensical value

  // upload the engineering
  boost::tie(outGol) = d_ops->uploadEngineering(d_extractLoBound,
                                                d_extractHiBound);

  // upload specific costs -- including increment the "shift" term
  d_ops->uploadShortrunCosts(d_dutySpecCosts, outGol);
  d_ops->uploadShortrunCosts(d_sizeSpecCosts * d_nameplateSize + d_standingCosts);

  // store duty values
  d_floorDuty   = d_extractLoBound;
  d_ceilingDuty = d_extractHiBound;

  // bind global cols to the relevant interfaces
  d_outCommodity->bindOsp(d_solver, outGol);

  // return the duty gol
  return outGol;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : washup
// ------------------------------------------------------------
//  Description  : washup call
//  Role         : obtain and process results
//  Techniques   : (nothing special)
//  Status       : complete
// ------------------------------------------------------------

template <typename C>
void
TeasSource<C>::washup()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, step", d_step);

  // results recovery
  double actualExtraction      = -1.0;       // nonsensical value
  boost::tie(actualExtraction) = d_ops->downloadSolution();

  // store entity state information
  d_extractions->at(d_step)    = actualExtraction;

  // short-run costs processing, outer call also updates instance data
  updateShortrunCosts
    (d_step,                                    // provided by class 'TicToc'
     d_ops->downloadShortrunCosts());

  // store some on-the-fly statistics
  d_dutyStats(actualExtraction);                // functor provided by class 'Block'
  d_sizeStats(d_extractHiBound);                // functor provided by class 'Block'
}

// ------------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ------------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// class 'Commodity' typedefs for convenience

typedef CmOxidize      Oxid;
```

```
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// class 'TeasLoad<>'

template TeasLoad<Oxid>::TeasLoad(const std::string, Record&);
template TeasLoad<Cert>::TeasLoad(const std::string, Record&);
template TeasLoad<Cseq>::TeasLoad(const std::string, Record&);
template TeasLoad<Elec>::TeasLoad(const std::string, Record&);
template TeasLoad<Work>::TeasLoad(const std::string, Record&);
template TeasLoad<Heat>::TeasLoad(const std::string, Record&);
template TeasLoad<Thrm>::TeasLoad(const std::string, Record&);
template TeasLoad<Fund>::TeasLoad(const std::string, Record&);

template TeasLoad<Oxid>::~TeasLoad();
template TeasLoad<Cert>::~TeasLoad();
template TeasLoad<Cseq>::~TeasLoad();
template TeasLoad<Elec>::~TeasLoad();
template TeasLoad<Work>::~TeasLoad();
template TeasLoad<Heat>::~TeasLoad();
template TeasLoad<Thrm>::~TeasLoad();
template TeasLoad<Fund>::~TeasLoad();

template const int TeasLoad<Oxid>::constrain(const xeona::DomainMode);
template const int TeasLoad<Cert>::constrain(const xeona::DomainMode);
template const int TeasLoad<Cseq>::constrain(const xeona::DomainMode);
template const int TeasLoad<Elec>::constrain(const xeona::DomainMode);
template const int TeasLoad<Work>::constrain(const xeona::DomainMode);
template const int TeasLoad<Heat>::constrain(const xeona::DomainMode);
template const int TeasLoad<Thrm>::constrain(const xeona::DomainMode);
template const int TeasLoad<Fund>::constrain(const xeona::DomainMode);

template void TeasLoad<Oxid>::washup();
template void TeasLoad<Cert>::washup();
template void TeasLoad<Cseq>::washup();
template void TeasLoad<Elec>::washup();
template void TeasLoad<Work>::washup();
template void TeasLoad<Heat>::washup();
template void TeasLoad<Thrm>::washup();
template void TeasLoad<Fund>::washup();

template shared_ptr<GateCom<Oxid> > TeasLoad<Oxid>::obtainGateway() const;
template shared_ptr<GateCom<Cert> > TeasLoad<Cert>::obtainGateway() const;
template shared_ptr<GateCom<Cseq> > TeasLoad<Cseq>::obtainGateway() const;
template shared_ptr<GateCom<Elec> > TeasLoad<Elec>::obtainGateway() const;
template shared_ptr<GateCom<Work> > TeasLoad<Work>::obtainGateway() const;
template shared_ptr<GateCom<Heat> > TeasLoad<Heat>::obtainGateway() const;
template shared_ptr<GateCom<Thrm> > TeasLoad<Thrm>::obtainGateway() const;
template shared_ptr<GateCom<Fund> > TeasLoad<Fund>::obtainGateway() const;

template shared_ptr<BandedTariffSet> TeasLoad<Oxid>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Cert>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Cseq>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Elec>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Work>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Heat>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Thrm>::obtainTariffSet() const;
template shared_ptr<BandedTariffSet> TeasLoad<Fund>::obtainTariffSet() const;

// 'TeasSource<>'

template TeasSource<Oxid>::TeasSource(const std::string, Record&);
template TeasSource<Cert>::TeasSource(const std::string, Record&);
template TeasSource<Cseq>::TeasSource(const std::string, Record&);
template TeasSource<Elec>::TeasSource(const std::string, Record&);
template TeasSource<Work>::TeasSource(const std::string, Record&);
template TeasSource<Heat>::TeasSource(const std::string, Record&);
template TeasSource<Thrm>::TeasSource(const std::string, Record&);
template TeasSource<Fund>::TeasSource(const std::string, Record&);

template TeasSource<Oxid>::~TeasSource();
template TeasSource<Cert>::~TeasSource();
template TeasSource<Cseq>::~TeasSource();
template TeasSource<Elec>::~TeasSource();
template TeasSource<Work>::~TeasSource();
template TeasSource<Heat>::~TeasSource();
```

```
template TeasSource<Thrm>::~TeasSource();
template TeasSource<Fund>::~TeasSource();

template const int TeasSource<Oxid>::constrain(const xeona::DomainMode);
template const int TeasSource<Cert>::constrain(const xeona::DomainMode);
template const int TeasSource<Cseq>::constrain(const xeona::DomainMode);
template const int TeasSource<Elec>::constrain(const xeona::DomainMode);
template const int TeasSource<Work>::constrain(const xeona::DomainMode);
template const int TeasSource<Heat>::constrain(const xeona::DomainMode);
template const int TeasSource<Thrm>::constrain(const xeona::DomainMode);
template const int TeasSource<Fund>::constrain(const xeona::DomainMode);

template void TeasSource<Oxid>::washup();
template void TeasSource<Cert>::washup();
template void TeasSource<Cseq>::washup();
template void TeasSource<Elec>::washup();
template void TeasSource<Work>::washup();
template void TeasSource<Heat>::washup();
template void TeasSource<Thrm>::washup();
template void TeasSource<Fund>::washup();

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : teasdev.cc
//  file-create-date : Thu 08-Oct-2009 13:00 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete technical assets development / implementation
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/teasdev.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  To be used for entity development.

//  LOCAL AND SYSTEM INCLUDES

#include "teasdev.h"          // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/label.h"       // helper class to format solver labels
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/optops.h"      // operate optimization sub-problems for hard assets
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// class 'Commodity' typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;
```

```
//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : tests.cc
//  file-create-date : Thu 24-Jan-2008 15:16 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : test sub-entities for use with --inbuilt and such / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/tests.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "tests.h"            // companion header for this file (place first)

#include "../c/conex.h"       // create and connect block interfaces
#include "../c/recset.h"      // record set support
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ------------------------------------------------------------
//  CLASS           : TestEntity0
// ------------------------------------------------------------
//  Purpose      : a representative first sub-class of FullEntity
//  Status       : working but incomplete
// ------------------------------------------------------------

// CREATORS

TestEntity0::TestEntity0
(const std::string entityId,                // enforced unique identifier
 Record&           record) :
  FullEntity(entityId, record)
{
  s_logger->repx(logga::dbug, "constructor call", "");
}

TestEntity0::~TestEntity0()
{
  s_logger->repx(logga::xtra, "destructor call", "");
}

void
TestEntity0::factoryInitialize()
{
  s_logger->repx(logga::xtra, "TestEntity0 instance initialization", "");
}

// ------------------------------------------------------------
//  CLASS           : TestEntity1
```

```
// ------------------------------------------------------------
//  Purpose      : a representative first sub-class of FullEntity
//  Status       : working but incomplete
//
//  Design notes
//
//      See notes for the superclass Entity.
//
//  CAUTION -- creating and initializing TestEntity1 objects
//
//      A TestEntity1 object MUST be instantiated using a shared
//      pointer, either:
//
//          shared_ptr<Entity>
//          shared_ptr<TestEntity1>
//
//      This is because the 'init' member function generates its
//      own shared pointer using function 'shared_from_this'.
//      The restriction is unlikely to be an issue outside of
//      testing, because the entity factory generates shared
//      pointers.
//
//      By way of background, 'shared_from_this' requires at
//      least one other shared pointer to be in existence.  The
//      Boost documentation, found at:
//
//          file:///usr/share/doc/libboost-doc/HTML/libs/
//          smart_ptr/enable_shared_from_this.html
//
//      says that the 'shared_from_this' member function
//      (actually there are two, depending on constness)
//      "requires: .. there must exist at least one shared_ptr
//      instance .. that owns .. [the underlying object]"
//
//      If a TestEntity1 object is created without producing a
//      shared pointer, calling 'init' will result in the
//      following kind of run-time error:
//
// 226  entity.cc   initialize   00  00.0000s   dbug   TestEntity1 instance initialization
// terminate called after throwing an instance of 'std::tr1::bad_weak_ptr'
//   what():  tr1::bad_weak_ptr
// /hom/.../scripts/mach: line 1308: 30080 Aborted (core dumped) ./entity
//
// ------------------------------------------------------------

//  STATIC DEFINITIONS

int TestEntity1::s_ctorCount = 0;                       // with explicit initialization
std::list<const TestEntity1*> TestEntity1::s_census;    // without explicit initialization

// CREATORS

TestEntity1::TestEntity1
(const std::string entityId,                 // enforced unique identifier
 Record&           record) :
  FullEntity(entityId, record),

  d_userDescription(record.tieSingle<std::string>("user-description")),
  d_x(record.tieSingle<double>("x")),
  d_single(record.tieSingle<bool>("single")),
  d_timeseries(record.tieTimeseries<int>("timeseries")),

  d_electricalOutput(record.tieTimeseries<int>("electrical-output")),
  d_runTime(record.tieSingle<double>("run-time")),
  d_someState(record.tieTimeseries<bool>("some-state"))
{
  s_logger->repx(logga::dbug, "constructor call", "");
  TestEntity1::s_census.push_back(this);      // insert copy of pointer in instance census
  s_logger->repx(logga::dbug, "current population", popnSize());

  d_builtinRemark = "beta";

  // a little modification to fake some real recalculation
  const unsigned steps = Entity::getHorizonSteps();
  d_electricalOutput->at(steps-1) = 0;
}

TestEntity1::~TestEntity1()
{
  s_logger->repx(logga::xtra, "destructor call", "");
  TestEntity1::s_census.remove(this);
```

```
}

// for future reference (above), note the safe census code was:
// TestEntity1::s_safeCensus.remove(static_pointer_cast<TestEntity1>(shared_from_this()));

void
TestEntity1::factoryInitialize()
{
  // initial reporting
  s_logger->repx(logga::xtra, "TestEntity1 instance initialization", "");

  // a little modification to fake some real recalculation
  const unsigned steps = Entity::getHorizonSteps();
  d_builtinRemark = "some built in remark";
  for ( unsigned i = 0; i < steps; ++i) d_electricalOutput->at(i) = i * 2;
  d_runTime = 123456;                       // +1.23e+05
  bool load = true;
  if ( getIdentifier() == "testme-two") load = false;
  for ( unsigned i = 0; i < steps; ++i) d_someState->at(i) = load;
  d_someState->resize(steps, true);
}

// UTILITY FUNCTIONS

// STREAM INSERTION SUPPORT

std::ostream&
TestEntity1::streamOut                      // support for overloaded operator<<
(std::ostream& os) const
{
  std::ios_base::fmtflags prior = os.flags();
  os << "    dummy output from a derived entity" << "\n";
  os.flags(prior);
  return os;
}

// STATIC ACCESSORS

const std::list<const TestEntity1*>&        // return a const reference
TestEntity1::getCensus()
{
  return s_census;
}

int
TestEntity1::popnSize()
{
  return s_census.size();
}

std::string                                 // returns string for test purposes
TestEntity1::traverseFullPopulation()
{
  std::ostringstream ss;
  ss << "traversal : |";
  BOOST_FOREACH( const TestEntity1* s, s_census )
    {
      ss << " " << s->getIdentifier() << " |";
    }
  ss << "\n";
  return ss.str();
}

// ------------------------------------------------------------
//  CLASS           : TestEntity2
// ------------------------------------------------------------
//  Purpose      : a representative first sub-class of FullEntity
//  Status       : working but incomplete
//
//  Design notes
//
//      Contains a 'Socket'
//
// ------------------------------------------------------------

//  STATIC DEFINITIONS

// CREATORS

TestEntity2::TestEntity2
(const std::string entityId,                // enforced unique identifier
```

```
 Record&           record) :
  FullEntity(entityId, record),
  d_socketWork(Socket<CmWork>::create(entityId, "one", "commodity-id"))
{
  s_logger->repx(logga::dbug, "constructor call", "");
  d_builtinRemark = "beta / with 'socket'";
}

TestEntity2::~TestEntity2()
{
  s_logger->repx(logga::xtra, "destructor call", "");
}

void
TestEntity2::factoryInitialize()
{
  s_logger->repx(logga::xtra, "TestEntity2 instance initialization", "");
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : tictoc.cc
//  file-create-date : Thu 13-Nov-2008 20:26 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : inherited interface for entities using common calls / implementat_n
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/b/tictoc.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "tictoc.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  CODE

// ------------------------------------------------------------
//  CLASS           : TicToc
// ------------------------------------------------------------
//  Description  : provide a call interface for common calls
//  Role         : to support a consistent usage across hosts classes
//  Techniques   : multiple inheritance, call to 'xeona::isTwoContained'
//  Status       : incomplete
//
//  Design notes
//
//      The 'TicToc' class provides a consistent simulation call
//      interface.  The class is inherited by the following
//      entities (noting that some of the sub-classes may not yet
//      actually exist):
//
//          'Block' sub-classes { 'TechnicalAsset' 'Gateway' 'Multibus' 'Environment' }
//          'Actor' sub-classes { 'AssetOperator' } but not 'DomainController'
//
// ------------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger TicToc::s_logger_tictoc = logga::ptrLogStream();

// CREATORS
```

```
TicToc::TicToc
(const int commitmentModeSum) :
  d_commitmentMode(xeona::e_modeNotSpecified),
  d_commitmentModeSum(commitmentModeSum),
  d_step(-1),                                 // nonsensical value
  d_solver()                                  // empty smart pointer
{
  // CAUTION: good to log the commitment strategy sum at the debug threshold
  s_logger_tictoc->repx(logga::dbug,
                        "constructor call, commit mode sum",
                        d_commitmentModeSum);
}

TicToc::~TicToc()
{
  s_logger_tictoc->repx(logga::adhc, "destructor call", "");
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : restructure
// ------------------------------------------------------------

void
TicToc::restructure
(const xeona::DomainMode commitmentMode)
{
  // initial reporting
  s_logger_tictoc->repx(logga::dbug, "entering member function", "");
  std::ostringstream put;
  put << "  restructure call arguments"                          << "\n"
      << "  commitment mode        : " << commitmentMode         << "\n";
  s_logger_tictoc->putx(logga::adhc, put);

  // update data members
  d_commitmentMode = commitmentMode;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : initialize
// ------------------------------------------------------------

void
TicToc::initialize
(const int                  step,
 shared_ptr<svif::SolverIf> solver)
{
  // initial reporting
  s_logger_tictoc->repx(logga::dbug, "entering member function", "");
  std::ostringstream put;
  put << "  initialize call arguments"                           << "\n"
      << "  step                   : " << step                   << "\n"
#ifndef _XUTEST // to avoid the need to link to unit 'siglp' when unit testing
      << "  solver optional tag    : " << solver->getProblemTag() << "\n"
#endif // _XUTEST
      << "  solver (pointer value) : " << solver                 << "\n";
  s_logger_tictoc->putx(logga::adhc, put);

  // update data members
  d_step           = step;
  d_solver         = solver;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : checkCommitmentMode
// ------------------------------------------------------------
//  Description  : tests for pure/sum miscibility and produces interpreted logging
//  Role         : provide sub-classes with a simple test
//  Techniques   : relies on 'xeona::isTwoContained'
//  Status       : complete
//
//  Design notes
//
//      Usage example
//
//          if ( ! checkCommitmentMode( d_commitmentMode, d_commitmentModeSum )
//          {
//            s_logger->repx(logga::warn, "commitment mode problem", "");
//          }
//
// ------------------------------------------------------------
```

```
bool                                        // 'false' is problematic
TicToc::checkCommitmentMode                 // non-virtual function
(const xeona::DomainMode pure,              // value under test
 const xeona::DomainMode sum)               // miscible values aggregate
{
  // establish input details for later reporting
  std::ostringstream oss;
  oss << pure << " : " << sum;

  // short circuit if test value not specified
  if ( pure == xeona::e_modeNotSpecified )
    {
      s_logger_tictoc->repx(logga::warn, "commitment unspecified, test : sum", oss.str());
      return false;
    }

  // call test function
  const tribool ret = xeona::isTwoContained(pure, sum);

  // CAUTION: three-state testing of a 'boost::logic::tribool'
  // value is NOT intuitive -- in particular, see the relevant
  // documentation file 'DOCS/CODE.txt'

  if ( ret )                                // 'true'
    {
      s_logger_tictoc->repx(logga::adhc, "miscible commitment, test : sum", oss.str());
      return true;
    }
  else if ( !ret )                          // 'false' means commitment immiscibility
    {
      s_logger_tictoc->repx(logga::dbug, "immiscible commitment, test : sum", oss.str());
      return false;
    }
  else                                      // 'indeterminate' means data problem
    {
      s_logger_tictoc->repx(logga::dbug, "data problem, test : sum", oss.str());
      return false;
    }
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : conex.cc
//  file-create-date : Wed 16-Jul-2008 15:22 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : create and connect block interfaces / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/conex.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "conex.h"            // companion header for this file (place first)

#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../b/optprob.h"     // optimization sub-problem and key sub-classes
#include "../b/commods.h"     // commodities hierarchy

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <typeinfo>           // run-time type information (RTTI)

#include <cctype>             // C-style char classification, case conversion

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro

//  CODE

// ----------------------------------------------------------
//  CLASS           : Interface (abstract base class)
// ----------------------------------------------------------
//  Description  : base class for 'Cable<C>' and 'Socket<C>' class templates
//  Role         : provide common static support for interface instantiations
//  Techniques   : 'weak_ptr', 'shared_ptr' casting, inheritance
//  See also     : 'Cable<C>' and 'Socket<C>'
//  Status       : complete
// ----------------------------------------------------------
//
//  Design notes
//
//      The rationale for this abstract base class
//
//          Templates take priority over static members --
//          therefore, this abstract base class (ABC) provides
//          static support before the templates are applied to
//          create the required sub-classes.
//
//      Weak pointers
//
```

```
//          'weak_ptr' smart pointers are used in the 's_census*'
//          static data members of type
//          'std::vector<weak_ptr<Interface> >', in order that
//          the reference count remains correct.
//
//          Karlsson (2006 p49) writes ".. when you have a
//          weak_ptr that's observing some resource, you'll
//          eventually want to access that resource.  To do so,
//          the weak_ptr must be converted to a shared_ptr,
//          because the weak_ptr alone does not allow direct
//          access to the resource."
//
//          A weak pointer is created from a shared pointer.
//
//              weak_ptr<T>(sp)        construction from shared_ptr<T>
//
//          There are two ways of creating a shared_ptr from a
//          weak_ptr, noting that implicit template type
//          conversions can also apply:
//
//              shared_ptr<T> sp(wp);          // [1]
//              shared_ptr<T> sp = wp.lock();  // [2]
//
//            note 1 : throws 'std::tr1::bad_weak_ptr' if wp is empty
//            note 2 : sp is empty if wp is empty
//
//      New commodities
//
//          Two modifications are required should new commodity
//          sub-classes need to be added:
//
//              extend cast list in 'connectAll'
//              extend explicit template instantiations list
//
//  See also
//
//      The design notes for class templates 'Cable<C>' and
//      'Socket<C>'.
//
//  References
//
//      Becker, Pete.  2007.  The C++ Standard Library extensions :
//        a tutorial and reference.  Addison-Wesley, Upper Saddle
//        River, New Jersey, USA.  ISBN 0-321-41299-0.
//
//      Karlsson, Bjoern.  2006.  Beyond the C++ Standard Library :
//        an introduction to Boost.  Addison-Wesley, Upper Saddle
//        River, New Jersey, USA.  ISBN 0-321-13354-4.
//
//      Lischner, Ray.  2003.  C++ in a nutshell : a language and
//        library reference.  O'Reilly and Associates, Sebastopol,
//        California, USA.  ISBN 0-596-00298-X.
//
// ------------------------------------------------------------

// STATIC DEFINITIONS

int Interface::s_interfaceCount      = 0;
int Interface::s_connectionCount     = 0;
int Interface::s_connectAllCallCount = 0;

std::vector<weak_ptr<Interface> > Interface::s_censusSockets;
std::vector<weak_ptr<Interface> > Interface::s_censusCables;

logga::spLogger Interface::s_logger  = logga::ptrLogStream(); // bind logger on definition

// CREATORS

// ------------------------------------------------------------
//  MEMBER FUNCTION : Interface
// ------------------------------------------------------------

Interface::Interface
(const std::string& hostId,
 const std::string& partnerId,              // empty string for sockets
 const std::string& ifQualifier,
 const std::string& commodityId) :
  d_hostId(hostId),                         // host is known at construction time
  d_partnerId(partnerId),                   // partner is known by cables only
  d_ifQualifier(ifQualifier),               // interface qualifier
  d_commodityId(commodityId),               // a common commodity is required
```

```
  d_myKey(),
  d_counterKey(),
  d_gol1(-1),                                // nonsensical value
  d_connected(false),                        // interface is unbound initially
  d_commodity(),                             // contains intensities, linked on connection
  d_flow(new double(0.0)),                   // the characterizing extensity
  d_cnn(new ConnectionOsp())                 // real shared pointer, but OSP lacks solver
{
  // initial reporting
  s_logger->repx(logga::dbug, "constructor call, host id", d_hostId);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~Interface
// ----------------------------------------------------------

Interface::~Interface()
{
  // initial reporting
  s_logger->repx(logga::xtra, "destructor call", "");
}

// FUNCTIONS FOR OPTIMIZATION SUB-PROBLEMS (OSP)

// ----------------------------------------------------------
//  MEMBER FUNCTION : bindOsp
// ----------------------------------------------------------
//  Description  : map OSP variables to block interfaces and then to solver
//  Role         : used within derived 'TechnicalAsset::constrain' calls
//  Techniques   : 'xeona::assign_ptr::revamp'
//  Status       : complete
// ----------------------------------------------------------

bool                                         // 'true' if OSP constraint is completed
Interface::bindOsp                           // map OSP variables to block interfaces
(shared_ptr<svif::SolverIf> solver,          // current solver interface instance
 const int                  gol)             // global col from relevant OSP call
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, gol", gol);

  // defensive programming
  if ( ! solver )
    {
      s_logger->repx(logga::warn, "solver pointer empty/null", solver);
    }
  if ( ! d_cnn )                             // should not get here
    {
      s_logger->repx(logga::warn, "ConnectionOsp pointer empty/null", d_cnn);
    }

  // process pass count, which is stored in the common
  // optimization sub-problem object -- the 'solver' argument is
  // simply used as an integrity check
  d_cnn->incPassCount(solver);               // initially zero
  const int pass = d_cnn->getPassCount();

  // can be used to visually confirm the solver is the same
  s_logger->repx(logga::adhc, "solver address", solver);

  // active code
  if ( pass == 1 )                           // first pass
    {
      // store 'gol' on first pass
      s_logger->repx(logga::adhc, "first pass, stored gol", gol);
      d_cnn->storeGol(gol);                  // store in volatile object
      return false;
    }
  else if ( pass == 2 )                      // second pass
    {
      // reestablish connection OSP on second pass, then process
      s_logger->repx(logga::adhc, "second pass, given gol", gol);

      const int gol1 = d_cnn->recoverGol();
      const int gol2 = gol;

      std::ostringstream put;
      put << "  hollow connection OSP pointer : " << d_cnn.get() << "\n"; // [1]

      // CAUTION: [1] 'get' required, else 'assign_ptr' streams
      // as 'true' or "1" (the reason being an incomplete
```

```
      // implementation of 'assign_ptr')

      // pool replace the hollow object with a full object
      s_logger->repx(logga::adhc, "about to revamp", "");
      d_cnn.revamp(new ConnectionOsp(solver, // real instance
                                     0));    // make pass zero (the default in any case)

      put << "  live connection OSP pointer   : " << d_cnn.get() << "\n";
      s_logger->repx(logga::adhc, "reestablish connection OSP complete", "");
      s_logger->putx(logga::adhc, put);

      // bind call
      return d_cnn->bindGols(gol1, gol2);
    }
  else                                      // should not get here
    {
      s_logger->repx(logga::warn, "pass count not {1,2}", pass);
      return false;
    }
}

// FUNCTIONS FOR HOST SUPPORT

// ----------------------------------------------------------
//  MEMBER FUNCTION : getPartner
// ----------------------------------------------------------

shared_ptr<Entity>
Interface::getPartner()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  if ( d_connected )
    {
      return Entity::retSharedPtr(d_partnerId);
    }
  else
    {
      s_logger->repx(logga::dbug, "returning empty shared pointer", d_partnerId);
      return shared_ptr<Entity>::shared_ptr();
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getCm
// ----------------------------------------------------------

shared_ptr<Commodity>
Interface::getCm()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  return d_commodity;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : tieFlow (overloaded)
// ----------------------------------------------------------

double&                                 // bind to reference or assign directly
Interface::tieFlow()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  return *d_flow;
}

void
Interface::tieFlow
(double& flow)                          // set by argument
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  *d_flow = flow;
}

// ----------------------------------------------------------
```

```
//  MEMBER FUNCTION : sayFlow
// ---------------------------------------------------------

void
Interface::sayFlow()                          // for testing purposes
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  std::ostringstream put;
  put << "  flow (interface " << d_myKey << ")"
      << " = " << *d_flow << "\n";
  s_logger->putx(logga::dbug, put);
}

// ---------------------------------------------------------
//  MEMBER FUNCTION : sayAll
// ---------------------------------------------------------

void
Interface::sayAll                             // for testing purposes
(const std::string& name)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  std::ostringstream put;
  put << std::boolalpha;
                        put << "  interface details"                          << "\n";
  if ( ! name.empty() ) put << "    local name (supplied)  : " << name          << "\n";
                        put << "    key                    : " << d_myKey       << "\n"
                            << "    connected              : " << d_connected   << "\n";
  if ( d_connected )    put << "    counter key            : " << d_counterKey  << "\n";
                        put << "    commodity id           : " << d_commodityId << "\n"
                            << "    current flow           : " << *d_flow       << "\n";
  s_logger->putx(logga::dbug, put);
}

// CONNECTIVITY FUNCTIONS

// ---------------------------------------------------------
//  MEMBER FUNCTION : connectAll (static)
// ---------------------------------------------------------
//  Description  : point of entry for making connections
//  Role         : loops cables, calls 'cableCast<C>' under various instantiations
//  Techniques   : static
//  Status       : complete
// ---------------------------------------------------------
//
//  Design notes
//
//      See code for treatment of multiple calls and of cables
//      which are already connected.  Depending on the
//      hash-conditional settings, both can trigger a warning
//      message and premature return.
//
// ---------------------------------------------------------

bool
Interface::connectAll()
{
  // initial reporting
  s_logger->repx(logga::info, "entering static member function", "");

  // preamble
  std::ostringstream put;
  bool ret = true;                            // return 'true' unless something fails

  // check call count -- and abandon if so coded and as required
  s_connectAllCallCount++;                    // initialized to zero
  if ( s_connectAllCallCount != 1 )
    {
      s_logger->repx(logga::warn, "repeat call, count", s_connectionCount);

#if 0 // 0 = allow multiple calls (to enable structural change), 1 = enforce single call
      return false;
#endif // 0

    }

  // integrity checks
```

```
  const int numCables  = s_censusCables.size();
  const int numSockets = s_censusSockets.size();

  std::ostringstream oss;
  oss << numCables << " : " << numSockets;
  s_logger->repx(logga::dbug, "cables : sockets", oss.str());

  if ( numCables != numSockets )
    {
      s_logger->repx(logga::warn, "cable/socket imbalance", oss.str());
      std::ostringstream oss;
      put << "  cable/socket imbalance detected"                              << "\n"
          << "    imbalance    : " << oss.str()                               << "\n"
          << "    explanation  : faulty model"                                << "\n"
          << "    likely cause : omitted or incorrect interface connections"  << "\n";
      s_logger->putx(logga::xtra, put);
    }

  // loop thru the cables
  int loopCount = 0;
  BOOST_FOREACH( weak_ptr<Interface> w, s_censusCables )
    {
      ++loopCount;                              // one-based counting
      s_logger->addSmartBlank(logga::xtra);
      put << "loop " << loopCount << "\n";
      s_logger->putx(logga::xtra, put);

      // check weak pointer
      if ( w.expired() )
        {
          s_logger->repx(logga::dbug, "expired weak pointer, cable index", loopCount);
          continue;
        }

      // attempt to recover a working pointer
      shared_ptr<Interface> s = w.lock();    // 'lock' will create empty from empty

      if ( s.get() == 0 )                    // empty or holding null
        {
          if ( s.use_count() == 0 )          // empty
            {
              s_logger->repx(logga::dbug, "empty shared pointer, cable index", loopCount);
            }
          else                               // therefore holding null
            {
              s_logger->repx(logga::dbug, "null shared pointer, cable index", loopCount);
            }
          ret = false;
          continue;                          // loop again
        }

      // skip cables which are already connected OR abandon task
      if ( s->isConnected() )
        {
          s_logger->repx(logga::warn, "cable already connected", s->d_myKey);

#if 0 // 0 = loop again, 1 = abandon task
          s_logger->repx(logga::warn, "abandoning task", "");
          return false;                      // abandon task
#else
          continue;                          // loop again
#endif // 0

        }

      // attempt to cast against a hardcoded list (any new
      // sub-commodities should therefore be added here)
      //
      // the 'cableCast' call, upon success, in turn calls to
      // 'makeConnection'

      if ( Interface::cableCast<CmOxidize>(s)      ) continue;
      if ( Interface::cableCast<CmCarbonSeq>(s)    ) continue;
      if ( Interface::cableCast<CmCarbonCert>(s)   ) continue;
      if ( Interface::cableCast<CmElectricity>(s)  ) continue;
      if ( Interface::cableCast<CmWork>(s)         ) continue;
      if ( Interface::cableCast<CmHeat>(s)         ) continue;
      if ( Interface::cableCast<CmThermalFluid>(s) ) continue;
      if ( Interface::cableCast<CmFunds>(s)        ) continue;

      // failure and function end reporting
```

```
      std::ostringstream oss;
      oss << "more details if report " << logga::xtra;
      s_logger->repx(logga::warn, "all 'cableCast' calls failed", oss.str());

      std::ostringstream put;
      put << "  commodity casting problem"                                    << "\n"
          << "    likely cause, given the key was found : sought commodity is not on"
          << " the hardcoded 'Interface::cableCast<Commodity>'"               << "\n"
          << "      call list in function 'Interface::connectAll', rerun with report "
          << logga::xtra
          << " to view the currently supported casts"                         << "\n"
          << "    otherwise : disregard this message and fix earlier problems"  << "\n";
      s_logger->putx(logga::dbug, put);
      s_logger->repx(logga::dbug, "leaving member function, returning", "false");

      ret = false;

    } // BOOST_FOREACH

  return ret;                                 // 'true' if everything worked
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : cableCast <> (static)
// ----------------------------------------------------------
//  Description  : cast a 'shared_ptr<Interface>' cable and proceed if successful
//  Role         : the next step in the connect chain
//  Techniques   : static, 'dynamic_pointer_cast' for smart pointer downcasting
//  Status       : complete
// ----------------------------------------------------------
//
//  CAUTION: smart pointer casts
//
//      'shared_ptr's have their own const, static, and dynamic
//      casts, namely:
//
//          const_pointer_cast
//          static_pointer_cast
//          dynamic_pointer_cast
//
// ----------------------------------------------------------

template <typename C>
shared_ptr<Interface>                         // zero on bad cast, empty on locate fail
Interface::cableCast
(const shared_ptr<Interface>& interface)
{
  // initial reporting
  std::ostringstream put;
  put << "entered function 'cableCast<"
      << xeona::trimLeadingDigits(typeid(C).name()) << ">'";
  s_logger->putx(logga::xtra, put);

  // attempt cast based on supplied 'C'
  shared_ptr<Cable<C> > cable = dynamic_pointer_cast<Cable<C> >(interface);  // downcast
  if ( cable == 0 )                           // bad cast returns zero
    {
      put << " : bad cast" << "\n";
      s_logger->putx(logga::xtra, put);
      return cable;                           // zero, in other words
    }
  put << " : good cast" << "\n";
  s_logger->putx(logga::xtra, put);

  // attempt to locate socket
  std::string socketKey = cable->d_counterKey;
  std::string cableId   = cable->d_hostId;

  shared_ptr<Socket<C> > socket = locateSocket<C>(socketKey, cableId);
  if ( ! socket )                             // either zero or empty
    {
      put << "... unable to locate socket" << "\n";
      s_logger->putx(logga::xtra, put);
      return socket;
    }

  // attempt connection
  s_logger->repx(logga::adhc, "about to make connection using", "");
  put << "  cable pointer  : " << cable  << "\n"
      << "  socket pointer : " << socket << "\n";
  s_logger->putx(logga::adhc, put);
```

```
  // attempt connection
  makeConnection(cable, socket);

  return socket;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : locateSocket <> (static)
// ----------------------------------------------------------
//  Description  : attempt to locate a suitable socket
//  Role         : the next step in the connect chain
//  Techniques   : static, 'dynamic_pointer_cast' for smart pointer downcasting
//  Status       : complete
// ----------------------------------------------------------

template <typename C>                       // C is restricted to Commodity sub-classes
shared_ptr<Socket<C> >                      // zero on bad cast, empty on locate fail
Interface::locateSocket
(const std::string& socketKey,              // with/without dot-qualification
 const std::string& cableId)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering static member function", "");
  std::ostringstream put;
  put << "entered function 'locateSocket<"
      << xeona::trimLeadingDigits(typeid(C).name()) << ">"
      << "(" << socketKey << ", " << cableId << ")'" << "\n";
  s_logger->putx(logga::xtra, put);

  // preamble
  shared_ptr<Socket<C> > empty;             // used as a fail-to-find return value
  empty = typename shared_ptr<Socket<C> >::shared_ptr();

  // loop thru the sockets
  int loopCount = 0;
  BOOST_FOREACH( weak_ptr<Interface> w, s_censusSockets )
    {
      ++loopCount;

      // attempt to recover a working pointer
      shared_ptr<Interface> s = w.lock();    // 'lock' will create empty from empty
      if ( s.get() == 0 )
        {
          if ( s.use_count() == 0 )
            {
              const int temp = loopCount;
              s_logger->repx(logga::dbug, "empty shared pointer, socket index", temp);
            }
          else
            {
              s_logger->repx(logga::dbug, "null shared pointer, socket index", loopCount);
            }
          continue;                         // loop again
        }

      // attempt to cast against current template
      shared_ptr<Socket<C> > socket = dynamic_pointer_cast<Socket<C> >(s); // downcast
      if ( socket == 0 )
        {
          put << " : bad cast (" << s->d_myKey << ")" << "\n";
          s_logger->putx(logga::xtra, put);
          continue;                         // loop again
        }
      else
        {
          put << " : good cast 2";
          s_logger->putx(logga::xtra, put);

          std::string currentKey = socket->d_myKey;

          if ( socketKey == currentKey )    // simple string match
            {
              put << " : key match (" << socketKey << ")";
              s_logger->putx(logga::xtra, put);
              if ( socket->isConnected() )  // most probably a model error
                {
                  s_logger->repx(logga::warn, "connection already bound", s->d_partnerId);
                  return empty;             // return empty pointer
                }
              put << " : caller should connect" << "\n";
```

```
              s_logger->putx(logga::xtra, put);
              return socket;                // key found, return good pointer
            }
        }                                   // fall thru, key not found
    } // BOOST_FOREACH

  put << "\n";
  s_logger->putx(logga::xtra, put);
  s_logger->repx(logga::warn, "socket key not found", socketKey);
  put << "  socket key (repeated) : " << socketKey << "\n"; // repeat as 'repx' truncates
  s_logger->putx(logga::dbug, put);
  s_logger->addSmartBlank(logga::dbug);
  return empty;                             // return empty pointer
  }

// ----------------------------------------------------------
//  MEMBER FUNCTION : makeConnection <> (static)
// ----------------------------------------------------------
//  Description  : bind the extensity
//  Role         : the final step in the connect chain
//  Techniques   : static, copy assign one pointer to the other
//  Status       : complete
// ----------------------------------------------------------
//
//  Design notes
//
//      integrity checks : connectedness bools, commodity ids
//      extra functions  : link commodity
//      main function    : bind extensity
//      housekeeping     : update necessary details
//
// ----------------------------------------------------------

template <typename C>                       // remember specialization can be employed
bool
Interface::makeConnection
(shared_ptr<Cable<C>  >& cable,
 shared_ptr<Socket<C> >& socket)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering static member function", "");
  std::ostringstream put;
  put << "entered function 'makeConnection<"
      << xeona::trimLeadingDigits(typeid(C).name()) << ">'" << "\n";
  s_logger->putx(logga::xtra, put);

  // check the connectedness bools (which should be fine in any case)
  if ( cable->d_connected || socket->d_connected )
    {
      s_logger->repx(logga::warn, "abandoning task, prior connections", "");
      std::ostringstream put;
      put << "  connectivity status (probable model problem)"    << "\n"
          << "    cable key           : " << cable->d_myKey      << "\n"
          << "    socket key          : " << socket->d_myKey     << "\n"
          << "    cable connectivity  : " << cable->d_connected  << "\n"
          << "    socket connectivity : " << socket->d_connected << "\n";
      s_logger->putx(logga::dbug, put);
      return false;
    }

  // check the commodity ids (which could well differ)
  std::string cableComId  =  cable->d_commodityId;
  std::string socketComId = socket->d_commodityId;
  if ( cableComId != socketComId )          // simple string match
    {
      s_logger->repx(logga::warn, "abandoning task, commodity mismatch", "");
      std::ostringstream put;
      put << "  commodity ids do not match (model problem)"      << "\n"
          << "    cable key           : " << cable->d_myKey      << "\n"
          << "    socket key          : " << socket->d_myKey     << "\n"
          << "    cable commodity id  : " << cableComId          << "\n"
          << "    socket commodity id : " << socketComId         << "\n";
      s_logger->putx(logga::dbug, put);
      return false;
    }

  // link in by recovering the pointers
  cable->d_commodity  = static_pointer_cast<Commodity>(Entity::retSharedPtr(cableComId));
  socket->d_commodity = static_pointer_cast<Commodity>(Entity::retSharedPtr(socketComId));

  // complete the identity information
```

```
  socket->d_partnerId  = cable->d_hostId ;   // update socket information
  socket->d_counterKey = cable->d_myKey ;    // update socket information

  s_logger->repx(logga::dbug, "about to bind extensity and OSP", "");

  // bind the extensity and the optimization sub-problem
  cable->d_flow = socket->d_flow;            // KAZAAP!
  cable->d_cnn  = socket->d_cnn;             // WOOMPF!

  // reporting
  put << "  socket key                           : " << socket->d_myKey << "\n"
      << "  socket connection OSP pointer (d_cnn) : " << socket->d_cnn   << "\n"
      << "  socket commodity                     : " << socketComId     << "\n"
      << "  cable key                            : " <<  cable->d_myKey << "\n"
      << "  cable connection OSP pointer (d_cnn)  : " <<  cable->d_cnn   << "\n"
      << "  cable commodity                      : " <<  cableComId     << "\n";
  s_logger->repx(logga::adhc, "low priority reporting follows", "");
  s_logger->putx(logga::adhc, put);

  // housekeeping
  cable->setConnected();
  socket->setConnected();
  ++s_connectionCount;

  put << " : connection complete" << "\n";
  s_logger->putx(logga::xtra, put);

  s_logger->repx(logga::adhc, "leaving member function", "success");
  return true;
}

// FUNCTIONS PROVIDED FOR UNIT TESTING

// ------------------------------------------------------------
//  MEMBER FUNCTION : getMyKey
// ------------------------------------------------------------

std::string
Interface::getMyKey()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  return d_myKey;                          // always known
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : isConnected
// ------------------------------------------------------------

bool
Interface::isConnected()
{
  // CAUTION: initial reporting omitted because it interferes
  // with the cast reporting

  return d_connected;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : reset (static)
// ------------------------------------------------------------

void
Interface::reset()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering static member function", "");

  s_logger->repx(logga::dbug, "zeroing 'Interface' static data", "");

  s_censusCables.clear();
  s_censusSockets.clear();

  s_interfaceCount      = 0;
  s_connectionCount     = 0;
  s_connectAllCallCount = 0;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : getInterfaceCount (static)
```

```
// ----------------------------------------------------------

int
Interface::getInterfaceCount()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering static member function", "");

  return s_interfaceCount;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getConnectionCount (static)
// ----------------------------------------------------------

int
Interface::getConnectionCount()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering static member function", "");

  return s_connectionCount;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : isComplete (static)
// ----------------------------------------------------------

bool
Interface::isComplete()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering static member function", "");

  return ! (s_interfaceCount - 2 * s_connectionCount);
}

// UTILITY FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : setConnected
// ----------------------------------------------------------

void
Interface::setConnected()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  d_connected = true;
}

// ----------------------------------------------------------
//  CLASS           : Cable <>
//  CLASS           : Socket <>
// ----------------------------------------------------------
//  Description  : a block-to-block interface is a key 'xeona' abstraction
//  Role         : to provide type-safe connectivity while keeping data overheads low
//  Techniques   : restricted templates, static factory function, no data members
//  Status       : complete
// ----------------------------------------------------------
//
//  Design notes
//
//      First up, the concept and orientation of "demand flow"
//      needs to be stated.  Demand flow refers to the commodity
//      requests that are passed, by definition, from cable to
//      socket.  Negative demand flow is NOT SUPPORTED -- in
//      which case, bidirectional blocks need special treatment
//      (as discussed shortly).  'xeona' currently supports [kg],
//      [J], and [UOA] (units-of-account) as characterizing
//      extensities.  By convention, demand flow is sketched
//      right-to-left.
//
//      For completeness, "commodity flow", as represented by a
//      simultaneous reduction and increase in characterizing
//      extensity, is often, but NOT NECESSARILY, anti-parallel
//      to demand flow.  For instance, cooling services
//      characterized with enthalpy flow parallel.  In contrast,
//      exergy flow is anti-parallel in most, but NOT ALL, cases.
//
```

```
//      Some blocks, for instance, line HV transmission assets,
//      are bidirectional and need to be assigned an arbitrary
//      demand flow direction.  At the time of writing,
//      bidirectional assets will be handled with two interfaces
//      to each node.
//
//      The characterizing extensity need not be literal -- in
//      the sense that carbon certificates are naturally
//      quantified in [kg] although no mass transport takes place.
//
//      Socket and cable containing blocks automatically create
//      interface objects upon their own construction.
//
//      The static function 'Interface::connectAll' then
//      completes the connections.
//
//      The set of possible template instantiations is restricted
//      by the list provided in the implementation file.
//
//  Connection process (undertaken by the 'Interface' class)
//
//      Two interfaces are connected as shown.  Starting with:
//
//          Host<>-->Socket<>-->Commodity<--<>Cable<--<>Host
//
//      where <>--> indicates a UML aggregation "part of"
//                  relationship with navigability restricted
//                  as shown
//
//      At various states, confirm that:
//
//          - the same 'Commodity' id is used
//          - the'C' types match
//          - the connectedness bools are 'false'
//
//      Then set:
//
//          - from cable: d_flow = socket->d_flow
//          - the connectedness bools to 'true'
//
//      From a design perspective, the 'Commodity' contains any
//      intensive information and the 'flow' variable represents
//      the characterizing extensity.
//
//  Smart pointers and the static factory function
//
//      Smart pointers are fantastic but they cannot
//      (unfortunately) be created in constructors.  The
//      solution here is to use a static factory function --
//      as described in the Boost documentation:
//
//          Smart Pointers > Smart Pointer Programming
//          Techniques > Obtaining a shared_ptr (weak_ptr) to
//          this in a constructor
//
//      This material is available at:
//
//          file:///home/robbie/boost-build/boost_1_35_0/libs/
//          smart_ptr/sp_techniques.html#in_constructor
//
//          file:///usr/share/doc/libboost-doc/HTML/libs/
//          smart_ptr/sp_techniques.html#in_constructor
//
//      A quote from the cited documentation (Boost 1.35.0)
//
//          If 'X' is supposed to always live on the heap,
//          and be managed by a shared_ptr, use a static
//          factory function:
//
//              class X
//              {
//              private:
//                X() { ... }
//              public:
//                static
//                shared_ptr<X>
//                create()
//                {
//                  shared_ptr<X> px(new X);
//                  return px;
//                }
```

```
//              };
//
//  See also
//
//      The design notes for base class 'Interface'.
//
// --------------------------------------------------------

// CREATORS

// --------------------------------------------------------
//  MEMBER FUNCTION : Cable<C>::Cable
// --------------------------------------------------------

template <typename C>
Cable<C>::Cable                                 // template signature "<C>" is required here
(const std::string& hostId,                     // see factory function for explanations
 const std::string& partnerId,
 const std::string& ifQualifier,
 const std::string& commodityId) :
  Interface(hostId, partnerId, ifQualifier, commodityId)
{
  std::string typestr = xeona::trimLeadingDigits(typeid(C).name());
  std::ostringstream put;
  put << "  cable constructor call"                          << "\n"
      << "    class         : Cable<"   << typestr   << ">" << "\n"
      << "    host id       : "         << hostId            << "\n"
      << "    partner id    : "         << partnerId         << "\n"
      << "    qualifier     : "         << ifQualifier       << "\n"
      << "    commodity id  : "         << commodityId       << "\n";
  s_logger->putx(logga::xtra, put);

  d_myKey = d_hostId;
  if ( ! d_ifQualifier.empty() ) d_myKey += "." + d_ifQualifier;
  d_counterKey = d_partnerId;
  if ( ! d_ifQualifier.empty() ) d_counterKey += "." + d_ifQualifier;
}

// --------------------------------------------------------
//  MEMBER FUNCTION : Cable<C>::create (static)
// --------------------------------------------------------

template <typename C>
shared_ptr<Cable<C> >
Cable<C>::create                                // sub-id form (added later)
(const std::string& hostId,                     // me
 const std::string& targetId_ifQualifier,       // form with dot-separated socket qualifer
 const std::string& commodityId)                // associated commodity
{
  // split the "sub-id" and confirm result
  std::string              targetId    = "";
  std::string              ifQualifier = "";
  std::vector<std::string> buffer;
  boost::split(buffer, targetId_ifQualifier, boost::is_any_of("."));
  if ( buffer.size() == 2 )
    {
      targetId    = buffer.at(0);
      ifQualifier = buffer.at(1);
    }
  else
    {
      s_logger->repx(logga::warn,
                     "sub-id not length two (check dot)",
                     targetId_ifQualifier);
    }

  // continue as per original (four argument) code
  shared_ptr<Cable<C> > pi(new Cable<C>(hostId, targetId, ifQualifier, commodityId));
  s_censusCables.push_back(weak_ptr<Cable<C> >(pi));
  ++s_interfaceCount;
  return pi;                                    // return the shared pointer
}

// --------------------------------------------------------
//  MEMBER FUNCTION : Socket<C>::Socket
// --------------------------------------------------------

template <typename C>                           // specializations can also be defined
Socket<C>::Socket                               // template signature "<C>" is required here
(const std::string& hostId,                     // see factory function for explanations
 const std::string& partnerId,
```

```
 const std::string& ifQualifier,
 const std::string& commodityId) :
  Interface(hostId, partnerId, ifQualifier, commodityId)
{
  std::string typestr = xeona::trimLeadingDigits(typeid(C).name());
  std::ostringstream put;
  put << "  socket constructor call"                        << "\n"
      << "    class         : Socket<"  << typestr   << ">" << "\n"
      << "    host id       : "         << hostId           << "\n"
      << "    partner id    : "         << partnerId        << "\n"
      << "    qualifier     : "         << ifQualifier      << "\n"
      << "    commodity id  : "         << commodityId      << "\n";
  s_logger->putx(logga::xtra, put);

  d_myKey = d_hostId;
  if ( ! d_ifQualifier.empty() ) d_myKey += "." + d_ifQualifier;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : Socket<C>::create (static)
// ------------------------------------------------------------

template <typename C>
shared_ptr<Socket<C> >
Socket<C>::create
(const std::string& hostId,                  // my id
 const std::string& ifQualifier,             // my socket qualifier
 const std::string& commodityId)             // associated commodity
{
  shared_ptr<Socket<C> > pi(new Socket<C>(hostId, "", ifQualifier, commodityId));
  s_censusSockets.push_back(weak_ptr<Socket<C> >(pi));
  ++s_interfaceCount;
  return pi;                                 // return the shared pointer
}

// ------------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ------------------------------------------------------------

// CAUTION: explicit template instantiations must be placed at
// the very end of the implementation file
//
// For more information, see the excellent documentation in
// 'recset.cc' regarding explicit template instantiations.

// Commodity typedefs for convenience

typedef CmOxidize      Oxid;
typedef CmCarbonCert   Cert;
typedef CmCarbonSeq    Cseq;
typedef CmElectricity  Elec;
typedef CmWork         Work;
typedef CmHeat         Heat;
typedef CmThermalFluid Thrm;
typedef CmFunds        Fund;

// instantiations

template Cable<Oxid>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Cseq>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Cert>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Elec>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Work>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Heat>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Thrm>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Cable<Fund>::Cable
  (const std::string&, const std::string&, const std::string&, const std::string&);

template shared_ptr<Cable<Oxid> > Cable<Oxid>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Cable<Cseq> > Cable<Cseq>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Cable<Cert> > Cable<Cert>::create
  (const std::string&, const std::string&, const std::string&);
```

```
template shared_ptr<Cable<Elec> > Cable<Elec>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Cable<Work> > Cable<Work>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Cable<Heat> > Cable<Heat>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Cable<Thrm> > Cable<Thrm>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Cable<Fund> > Cable<Fund>::create
  (const std::string&, const std::string&, const std::string&);

template Socket<Oxid>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Cseq>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Cert>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Elec>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Work>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Heat>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Thrm>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);
template Socket<Fund>::Socket
  (const std::string&, const std::string&, const std::string&, const std::string&);

template shared_ptr<Socket<Oxid> > Socket<Oxid>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Cseq> > Socket<Cseq>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Cert> > Socket<Cert>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Elec> > Socket<Elec>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Work> > Socket<Work>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Heat> > Socket<Heat>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Thrm> > Socket<Thrm>::create
  (const std::string&, const std::string&, const std::string&);
template shared_ptr<Socket<Fund> > Socket<Fund>::create
  (const std::string&, const std::string&, const std::string&);

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : costs.cc
//  file-create-date : Fri 17-Oct-2008 12:00 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : cost sets and support / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/costs.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "costs.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>             // printf style formatting
#include <boost/tuple/tuple.hpp>        // n-tuples, ref(), cref()

//  CODE

// ---------------------------------------------------------
//  CLASS           : CostSet
// ---------------------------------------------------------
//  Description  : container for multiple cost types, be they intensive or extensive
//  Role         : to support 'OptimSubProb' instances and for cost aggregation
//  Techniques   : struct, overloaded operators, tuples
//  Status       : complete
//
//  Supported arithmetic and comparison operators
//
//          typedef tuple<double, double, double, double> tupleD5
//
//          add                             : cs1  = cs2 + cs3
//          subtract                        : cs1  = cs2 - cs3
//          add and assign                  : cs1 += cs2
//          subtract and assign             : cs1 -= cs2
//          scalar multiply                 : cs1  = double * cs2
//          scalar multiply                 : cs1  = cs2 * double
//          scalar multiply and assign      : cs1 *= double
//
//          tupleD5 multiply                : cs1  = tupleD5 * cs2
//          tupleD5 multiply                : cs1  = cs2 * tupleD5
//          tupleD5 multiply and assign     : cs1 *= tupleD5
//
//          strictly equal to (no epsilon)  : cs1 == cs2
//          strictly not equal to           : cs1 != cs2
```

```
//
//          Note also the 'tupleD5' can be used in constructors
//          and can be exported using the 'tuple' member
//          function.
//
//  Design notes
//
//      Public data
//
//          This class is indeed a 'struct'.  It has public data,
//          which means direct assignment and recovery is legal:
//
//              cs1.fin = 111.1;
//              std::cout << cs1.ghg << std::endl;
//
//      Free binary operators
//
//          The overloaded binary operators used here are
//          implemented as free functions.  They could equally be
//          member functions.  See Stephens etal (2006 p327) for
//          a discussion on the relative merits.
//
//          The free binary operators are not granted friendship
//          because they do not need it -- the data is public.
//
//      Copy constructor and copy assignment operator
//
//          The compiler supplied copy constructor and copy
//          assignment operator will do just fine:
//
//             CostSet(const CostSet& orig);              // copy constructor
//             CostSet& operator= (const CostSet& orig);  // copy assignment operator
//
//  References
//
//      Stephens, D Ryan, Christopher Diggins, Jonathan Turkanis,
//        and Jeff Cogswell.  2006.  C++ cookbook : solutions and
//        examples for C++ programmers.  O'Reilly Media,
//        Sebastopol, California, USA.  ISBN 0-596-00761-2.
//
// ------------------------------------------------------------

// CREATORS

CostSet::CostSet
(const double init) :                       // the default of zero is set in the header
  fin(init),
  ghg(init),
  nox(init),
  dep(init),
  luc(init)
{ }

CostSet::CostSet                            // up-front constructor
(const double a_fin,
 const double a_ghg,
 const double a_nox,
 const double a_dep,
 const double a_luc) :
  fin(a_fin),
  ghg(a_ghg),
  nox(a_nox),
  dep(a_dep),
  luc(a_luc)
{ }

CostSet::CostSet                            // tuple constructor
(const tupleD5 tup) :
  fin(tup.get<0>()),
  ghg(tup.get<1>()),
  nox(tup.get<2>()),
  dep(tup.get<3>()),
  luc(tup.get<4>())
{ }

CostSet::~CostSet() { }                     // destructor

// UNARY OPERATORS

// Dattatri (2002) covers the design of overloaded operators in
// chapter 8.
```

```
//
// Dattatri, Kayshav.  2002.  C++ : effective-object oriented
//   software construction : concepts, principles, industrial
//   strategies and practices -- Second edition.  Prentice Hall
//   PTR, Upper Saddle River, New Jersey, USA.  ISBN
//   0-13-086769-1.

CostSet
CostSet::operator-() const
{
  CostSet cs;
  cs.fin = -fin;
  cs.ghg = -ghg;
  cs.nox = -nox;
  cs.dep = -dep;
  cs.luc = -luc;
  return cs;
}

CostSet&
CostSet::operator+=
(const CostSet& other)
{
  fin += other.fin;
  ghg += other.ghg;
  nox += other.nox;
  dep += other.dep;
  luc += other.luc;
  return *this;
}

CostSet&
CostSet::operator-=
(const CostSet& other)
{
  fin -= other.fin;
  ghg -= other.ghg;
  nox -= other.nox;
  dep -= other.dep;
  luc -= other.luc;
  return *this;
}

CostSet&
CostSet::operator*=
(const double& other)
{
  fin *= other;
  ghg *= other;
  nox *= other;
  dep *= other;
  luc *= other;
  return *this;
}

CostSet&
CostSet::operator*=
(const tupleD5& other)
{
  fin *= other.get<0>();
  ghg *= other.get<1>();
  nox *= other.get<2>();
  dep *= other.get<3>();
  luc *= other.get<4>();
  return *this;
}

// ACCESSORS

tupleD5
CostSet::tuple() const
{
  return boost::make_tuple(fin, ghg, nox, dep, luc);
}

bool
CostSet::isZero() const
{
  if ( fin    == 0.0
       && ghg == 0.0
       && nox == 0.0
```

```
       && dep == 0.0
       && luc == 0.0)
    return true;
  else
    return false;
}

// MANIPULATORS

void CostSet::reset(const double value)
{ fin = value; ghg = value; nox = value; dep = value; luc = value; }

void CostSet::setFin(const double setFin) { fin = setFin; }
void CostSet::setGhg(const double setGhg) { ghg = setGhg; }
void CostSet::setNox(const double setNox) { nox = setNox; }
void CostSet::setDep(const double setDep) { dep = setDep; }
void CostSet::setLuc(const double setLuc) { luc = setLuc; }

void CostSet::plusFin(const double plusFin) { fin += plusFin; }
void CostSet::plusGhg(const double plusGhg) { ghg += plusGhg; }
void CostSet::plusNox(const double plusNox) { nox += plusNox; }
void CostSet::plusDep(const double plusDep) { dep += plusDep; }
void CostSet::plusLuc(const double plusLuc) { luc += plusLuc; }

void CostSet::multFin(const double multFin) { fin *= multFin; }
void CostSet::multGhg(const double multGhg) { ghg *= multGhg; }
void CostSet::multNox(const double multNox) { nox *= multNox; }
void CostSet::multDep(const double multDep) { dep *= multDep; }
void CostSet::multLuc(const double multLuc) { luc *= multLuc; }

// DISPLAY CALLS

std::string
CostSet::summarizeMeG                          // general float format
(std::string msg) const
{
  if ( msg.empty() ) msg = "(no msg)";
  std::ostringstream oss;
  oss << boost::format("  %15s :") %  msg
      << boost::format("   fin = %10.3g") % fin
      << boost::format("   ghg = %10.3g") % ghg
      << boost::format("   nox = %10.3g") % nox
      << boost::format("   dep = %10.3g") % dep
      << boost::format("   luc = %10.3g") % luc;
  return oss.str();
}

std::string
CostSet::summarizeMeF                          // fixed float format
(std::string msg) const
{
  if ( msg.empty() ) msg = "(no msg)";
  std::ostringstream oss;
  oss << boost::format("  %15s :") %  msg
      << boost::format("   fin = %10.2f") % fin
      << boost::format("   ghg = %10.2f") % ghg
      << boost::format("   nox = %10.2f") % nox
      << boost::format("   dep = %10.2f") % dep
      << boost::format("   luc = %10.2f") % luc;
  return oss.str();
}

// ------------------------------------------------------
//  MEMBER FUNCTION : streamOut
// ------------------------------------------------------
//  Description  : workhorse for overloaded operator<<
//  Role         : for streaming cost sets
//  Techniques   : 'std::ostream'
//  Status       : working
// ------------------------------------------------------

std::ostream&
CostSet::streamOut                              // support for overloaded operator<<
(std::ostream& os) const
{
  std::ios::fmtflags prior = os.flags();      // grab ostream state
  os << boost::format(  "fin = %.3g") % fin
     << boost::format("  ghg = %.3g") % ghg
     << boost::format("  nox = %.3g") % nox
     << boost::format("  dep = %.3g") % dep
     << boost::format("  luc = %.3g") % luc;
```

```
  os.flags(prior);                          // reset ostream state
  return os;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator+ (CostSet&, CostSet&)
// ------------------------------------------------------------

const CostSet
operator+
(const CostSet& lhs,
 const CostSet& rhs)
{
  CostSet tmp;
  tmp.fin = lhs.fin + rhs.fin;
  tmp.ghg = lhs.ghg + rhs.ghg;
  tmp.nox = lhs.nox + rhs.nox;
  tmp.dep = lhs.dep + rhs.dep;
  tmp.luc = lhs.luc + rhs.luc;
  return tmp;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator- (CostSet&, CostSet&)
// ------------------------------------------------------------

const CostSet
operator-
(const CostSet& lhs,
 const CostSet& rhs)
{
  CostSet tmp;
  tmp.fin = lhs.fin - rhs.fin;
  tmp.ghg = lhs.ghg - rhs.ghg;
  tmp.nox = lhs.nox - rhs.nox;
  tmp.dep = lhs.dep - rhs.dep;
  tmp.luc = lhs.luc - rhs.luc;
  return tmp;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator* (double&,  CostSet&)
// ------------------------------------------------------------

const CostSet
operator*
(const double&  lhs,
 const CostSet& rhs)
{
  CostSet tmp;
  tmp.fin = lhs * rhs.fin;
  tmp.ghg = lhs * rhs.ghg;
  tmp.nox = lhs * rhs.nox;
  tmp.dep = lhs * rhs.dep;
  tmp.luc = lhs * rhs.luc;
  return tmp;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator* (CostSet&, double&)
// ------------------------------------------------------------

const CostSet
operator*
(const CostSet& lhs,
 const double&  rhs)
{
  return operator*(rhs, lhs);
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator* (CostSet&, tupleD5&)
// ------------------------------------------------------------

const CostSet
operator*
(const CostSet& lhs,
 const tupleD5& rhs)
{
  CostSet tmp;
  tmp.fin = lhs.fin * rhs.get<0>();
```

```
  tmp.ghg = lhs.ghg * rhs.get<1>();
  tmp.nox = lhs.nox * rhs.get<2>();
  tmp.dep = lhs.dep * rhs.get<3>();
  tmp.luc = lhs.luc * rhs.get<4>();
  return tmp;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator* (tupleD5&, CostSet&)
// ------------------------------------------------------------

const CostSet
operator*
(const tupleD5& lhs,
 const CostSet& rhs)
{
  return operator*(rhs, lhs);
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator== (CostSet&, CostSet&)
// ------------------------------------------------------------

bool
operator==
(const CostSet& lhs,
 const CostSet& rhs)
{
  return (lhs.fin == rhs.fin &&
          lhs.ghg == rhs.ghg &&
          lhs.nox == rhs.nox &&
          lhs.dep == rhs.dep &&
          lhs.luc == rhs.luc);
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator!= (CostSet&, CostSet&)
// ------------------------------------------------------------

bool
operator!=
(const CostSet& lhs,
 const CostSet& rhs)
{
  return ! (lhs == rhs);
}

// ------------------------------------------------------------
//  FREE FUNCTION   : operator- (boost::tuple<CostSet, CostSet>)
// ------------------------------------------------------------

boost::tuple <CostSet, CostSet>              // normally "var" and "fix" costs
operator-                                    // unary minus (not subtraction)
(const boost::tuple <CostSet, CostSet>& other)
{
  // reporting
  logga::spLogger logger = logga::ptrLogStream();
  logger->repx(logga::xtra, "tuple<CostSet, CostSet> unary minus", "");

  // active code
  return boost::make_tuple(-other.get<0>(),  // unary minus on recovered 'CostSet' object
                           -other.get<1>());
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : datio.cc
//  file-create-date : Mon 01-Oct-2007 12:24 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : model file read, process, and write functionality / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/datio.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "datio.h"            // companion header for this file (place first)

#include "../c/recset.h"      // record set support
#include "../b/entity.h"      // entity base class
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <fstream>            // file-based io
#include <iomanip>            // setw() and family
#include <locale>             // locale specific information
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/any.hpp>                // type heterogeneous storage
#include <boost/cast.hpp>               // numeric_cast<> number to number conversions
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

// LOCAL DEFINITIONS

// Set the size of the getline data retrieval buffer using the
// integer ::BUFFSIZE_GETLINE -- this can be extremely generous
// because it only persists for the data retrieval period (note
// that 33 * 8760 equals about 0.3MiB, assuming worst case 32
// char decimal numbers within an annual hourly time-series).

namespace // unnamed
{
#if 0 // 0 = safe, 1 = for testing with an inadequate buffsize and 'datrec.ut1.cc'
  const int BUFFSIZE_GETLINE = 150;   // at time of writing, needs 161 + 1 characters
# warning "getline buffer being deliberately undersized for test purposes"
#else
  const int BUFFSIZE_GETLINE = 1024 * 1024 / 1;   // 1.0MiB
#endif // 0
} // unnamed namespace

//  CODE
```

```
// ----------------------------------------------------------
//  CLASS           : DataIo
// ----------------------------------------------------------
//  Description  : management of model data input and output
//  Role         : initially read and later overwrite the XEM model file
//  Status       : complete
//
//  Text file formatting issues
//
//      record  : must have no leading whitespace
//      field   : should have (ideally four) leading whitespaces
//      comment : determined through a process of elimination
//
//  Design notes
//
//      'DataIo' is an unenforced singleton.
//
//      Whitespace characters for Boost.String_algo and STL libraries
//
//          The STL classification function 'std::isspace' is
//          locale-specific.  The Boost.String_algo
//          classification predicates (for instance,
//          'boost::is_space') simply wrap the functionality of
//          the STL classification functions (in this case
//          'std::isspace').  This means identical conclusions
//          must arise irrespective of which library calls are
//          made.
//
//          Hence, the value of 'std::locale' is used to
//          establish the whitespace chars for 'boost::trim_*'.
//
//          For the "C" locale, space characters comprise
//          (Lischner 2003 pp573-574) and are defined (Lischner
//          2003 p6) thus:
//
//              ' '     space
//              '\f'    form feed
//              '\n'    newline
//              '\r'    carriage return
//              '\t'    tab
//              '\v'    vertical tab
//
//          If problems arise, it may be necessary to run 'xeona'
//          under "LANG=C" or "LC_CTYPE=C" to enforce the above
//          definitions.
//
//      Possible design extensions
//
//          Note that 'xeona::e_end' could be treated as a
//          comments-only record kind.
//
//          In retrospect, I would have looked carefully at using
//          the 'Boost.Spirit' parser library for the processing
//          of XEM files.  This would probably have also required
//          an EBNF-style grammar specification for the XEM
//          structure.
//
//      Connectivity information
//
//          Connectivity information (for better or worse) is
//          currently embedded within entity specifications.
//
// ----------------------------------------------------------

//  STATIC DEFINITIONS

unsigned DataIo::s_tabset = xeona::modelAngleIndent;   // set in 'common.cc'

logga::spLogger DataIo::s_logger
  = logga::ptrLogStream();                  // bind logger on definition

// CREATORS

DataIo::DataIo
(RecordSet& recset) :
  d_records(recset),                        // bind by reference
  d_model(),                                // empty path
  d_curRecord(new Record()),                // empty current record object
  d_curField(new Field()),                  // empty current field object
  d_fieldWriteDepth(e_notSet),              // default
```

```
  d_lineCount(0),                           // zero line count
  d_recCount(0),                            // zero pushed-back record count
  d_flag_initialRecord(true),               // necessary for processing logic
  d_flag_endMarker(false)                   // necessary for integrate verification
{
  s_logger->repx(logga::dbug, "constructor call", "");
}

DataIo::~DataIo()
{
  s_logger->repx(logga::dbug, "destructor call", "");

  unsigned indent = xeona::modelFieldIndent; // set in 'common.cc'
  unsigned tabset = indent + s_tabset + 1;   // determined here
  std::ostringstream put;
  put << "    DataIo destructor call data"             << "\n";
  put << "    record-set count : " << getRecordCount() << "\n";
  put << "    indent (col)     : " << indent           << "\n";
  put << "    tabset (col)     : " << tabset           << "\n";
  s_logger->putx(logga::xtra, put);
}

// ACCESSORS

boost::filesystem::path
DataIo::getModelFilePath()
{
  return d_model;
}

unsigned
DataIo::getRecordCount()
{
  return d_records.getCount();
}

  // STATIC ACCESSORS

unsigned
DataIo::getTabset()
{
  return s_tabset;
}

// MANIPULATORS (public and with identifiers that contain the phrase "model")

// -----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::readModel
// -----------------------------------------------------------
//  Description  : open the file, cycle getlines, call 'loadDataLine'
//  Role         : public point of access for reading model into record set
//  Precondition : model path should be valid
//  Status       : complete
// -----------------------------------------------------------

void
DataIo::readModel
(boost::filesystem::path modelFile)          // model file path
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "");

  // process object name
  std::string filename  = modelFile.string();          // in operating system format [1]
  const char* cfilename = filename.c_str();            // as C-string for fstream objects

  // CAUTION: [1] boost::filesystem::native_file_string() is now depreciated
  // (despite still being present in the official demo files)

  // open file in read-only mode
  std::ifstream infile(cfilename);           // defaults to text and std::ios::in
  if ( ! infile )
    {
      std::cout << std::flush;
      std::clog
        << "\n"
        << "** unsuccessful ifstream construction on: " << cfilename << "\n"
        << "   (earlier filename and write status checks may have also failed)"
        << "\n"
        << std::endl;
    }
```

```
  while ( true )                            // for use with 'break' command
    {
      // for information on getline(), see Josuttis (1999 p608)
      // and Lischner (2003 p531) -- note also the char array
      // initialization, filled with null chars, as recommended
      // by Sutter and Alexandrescu (2005 p37)

      char cline[::BUFFSIZE_GETLINE] = {'\0'};    // C-style buffer for getline()
      infile.getline(cline, ::BUFFSIZE_GETLINE);  // read until newline or BUFFSIZE - 1

      // abandon loop if end of file
      if ( infile.eof() ) break;

      // for information on iostate literals (such as
      // 'std::ios::failbit'), see Lischner (2003 p517) -- and
      // for information on 'std::fstream' member functions (such
      // as 'rdstate'), see Lischner (2003 pp505-508)

      // check stream status, looking for 'failbit', but not 'badbit' or 'eofbit'
      if ( infile.rdstate() & std::ios::failbit )     // CAUTION: '&' is correct
        {
          std::cout << std::flush;
          std::clog
            << "\n"
            << "** unsuccessful getline() operation, std::ios::failbit set"
            << std::endl;
          std::cout << std::flush;
          std::clog
            << "** maximum line length may have exceeded character buffer capacity of : "
            << ::BUFFSIZE_GETLINE - 1
            << std::endl;

#ifdef __unix__ // UNIX-specific code, additional reporting only

          std::string shellCommand                    // ensure sufficient blanks
            = "wc --max-line-length < " + filename;    // std::string version of filename

          std::cout << std::flush;
          std::clog << "** " << shellCommand << " gives : " << std::flush;
          const int ret = system(shellCommand.c_str());
          std::clog << std::endl;                     // additional to 'system' newline
          if ( ret != 0 )                             // unlikely to trigger
            {
              s_logger->repx(logga::warn, "system (shell) returned nonzero", ret);
            }

#else
//  #  warning "UNIX-specific additional reporting being omitted"
#endif // __unix__

          s_logger->repx(logga::kill,
                         "getline() fail, undersize buffer?",
                         ::BUFFSIZE_GETLINE);
          return;
        };

      if ( infile.rdstate() & std::ios::badbit )     // CAUTION: '&' is correct
        {
          s_logger->repx(logga::kill, "getline() fail: system-level error?", "");
          std::cout << std::flush;
          std::clog
            << "\n"
            << "** unsuccessful getline() operation, std::ios::badbit set" << "\n"
            << "   (system-level input/output error?)"
            << "\n"
            << std::endl;
          return;
        };

      // CAUTION: note that due to the way 'getline' works, a
      // line has no final 'newline' char and therefore may be of
      // zero length

      // process the line
      if ( loadDataLine(cline) == false )    // principal call
        {
          s_logger->repx(logga::kill, "loadDataLine call failed", false);
          return;
        }
```

```
    } // on block exit, 'cline' releases its memory and 'infile' closes

  d_records.setStatus(RecordSet::e_loaded);  // update status
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::confirmModelSvnAlign
// ----------------------------------------------------------
//  Description  : obtain minimum model format and compare against svn revision
//  Role         : used to confirm model/application match
//  Precondition : in-sync svn revision (although not-in-sync tolerant)
//  Status       : tested
// ----------------------------------------------------------

bool                                    // 'true' is model format and svn rev align
DataIo::confirmModelSvnAlign()
{
  // check for not-in-sync svn
  const int currentSvn = xeona::svnRev;
  if ( currentSvn == 0 )                    // codebase not-in-sync
    {
      s_logger->repx(logga::dbug, "unable to confirm model/svn match", "");
      return false;
    }

  // obtain model data
  const shared_ptr<Field> f
    = d_records.locateRecordAndField("data-format", "minimum-svn");
  if ( ! f )
    {
      s_logger->repx(logga::dbug, "unable to confirm model/svn match", "");
      return false;
    }

  // check for mismatch
  const std::string minimumSvnStr = f->getRawStr();
  const int minimumSvn            = boost::lexical_cast<int>(minimumSvnStr);
  if ( minimumSvn > currentSvn )
    {
      std::stringstream oss1;
      oss1 << minimumSvn << " > " << currentSvn;
      s_logger->repx(logga::warn, "model/svn mismatch", oss1.str());
      return false;
    }
  else if ( minimumSvn == currentSvn )
    {
      std::stringstream oss2;
      oss2 << minimumSvn << " = " << currentSvn;
      s_logger->repx(logga::info, "model/svn match confirmed", oss2.str());
      return true;
    }
  else
    {
      std::stringstream oss3;
      oss3 << minimumSvn << " < " << currentSvn;
      s_logger->repx(logga::info, "model/svn backwards compatibility?", oss3.str());
      return true;
    }
}

// UTILITY FUNCTIONS (private and with identifiers that often contain the phrase "data")

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::loadDataLine
// ----------------------------------------------------------
//  Description  : process input and load records
//  Role         : called by DataIo::readModel()
//  Status       : mostly complete
//
//  Design notes
//
//      Parses each line on its merits alone and does not
//      consider its wider context, for instance, is it part of a
//      'note' record or an 'entity' record (I suppose this is a
//      feature).
//
//      Some of the record kinds supported here are now obsolete.
//      The fact that parsing code is still included should not
//      present a problem.  For example:
//
//          current  : note, program, entity, model-end
```

```
//
//      The output from option '--data' should yield the latest
//      data rules.
//
// ------------------------------------------------------------

bool                                        // returns 'true' on success
DataIo::loadDataLine
(const char* cline)                         // C-string input line (from getline)
{
  // PREAMBLE

  // update the line count
  ++d_lineCount;

  // create some buffers
  std::string              line;            // read in text line
  std::string              buffer;          // general buffer
  std::vector<std::string> split;           // boost::split() results

  // create and initialize some variables
  tribool           lineEnabled = indeterminate;       // 'true' indicates enabled
  xeona::RecordKind recordKind  = xeona::e_notRecord;  // thus record header if reset
  xeona::FieldKind  fieldKind   = xeona::e_notField;   // thus field line if reset

  // LOAD LINE INTO STRING BUFFER

  line = cline;                             // C-string to std::string conversion
  boost::trim_right(line);                  // remove any trailing whitespaces
  const std::string oline = line;           // trimmed original line for comments

  // DEAL WITH TAB CHARS : embedded tabs in XEM files are highly discouraged!

  if ( boost::find_first(line, "\t") )      // iterator_range is convertible to bool
    // CAUTION: must use 'boost::find_first' and not
    // 'boost::find' above (could also use 'boost::contains' predicate)
    {
      s_logger->repx(logga::warn, "tab found in line", d_lineCount);
      if ( ! xeona::DBUG )
        {
          boost::replace_all(line, "\t", " ");    // replace tabs if not _XDEBUG
        }
    }

  // DETERMINE DISABLED STATUS

  // look for and then note and remove the 'xeona::modelDisableChar'
  // disable character if present -- defined as a '#' at the time
  // of writing

  std::string padding;                      // any leading whitespace characters
  std::string content;                      // remainder of line
  content = boost::trim_left_copy(line);    // could be an empty string
  padding = boost::erase_tail_copy(line, content.length());

  if ( boost::starts_with(content, xeona::modelDisableChar) )
    {
      content = content.substr(1);          // disabled, remove leading char
      boost::trim_left(content);            // remove any intervening whitespaces
      line  = padding + content;            // copy back without disable character
      lineEnabled = false;                  // set status
    }
  else if ( ! content.empty() )
    {
      lineEnabled = true;
    }

  // DETERMINE THE LINE TYPE

  // a record must have no leading whitespace whereas a field
  // ideally should -- in addition, a comment is determined later
  // on through a process of elimination

  // CAUTION: read Josuttis (1999 pp495-496) VERY CAREFULLY if
  // you opt for STL string rather that Boost.String_algo search
  // functions -- particularly in regard to return types, failure
  // to find detection, and 'std::string::npos'!

  // first scan for record headers, 'fieldKind' remains zeroed
  if ( boost::equals(line, "note"))                   // equals() returns bool
    recordKind = xeona::e_note;
```

```
  else if ( boost::starts_with(line, "entity.") )      // starts_with() returns bool
    recordKind = xeona::e_entity;
  else if ( boost::starts_with(line, "program.") )
    recordKind = xeona::e_program;
  else if ( boost::starts_with(line, xeona::modelEndMarker) )  // defined in 'common.cc'
    recordKind = xeona::e_end;          // a special record kind containing only comments

  // and then scan for field lines, 'recordKind' remains zeroed
  else if ( boost::contains(line, ">") )     // contains() returns bool
    fieldKind = xeona::e_input;
  else if ( boost::contains(line, "<") )
    fieldKind = xeona::e_output;

  // then deal with the special case of comments containing both
  // '<' and '>' (perhaps an email address like: <name@provider>)
  if ( boost::contains(line, "<") && boost::contains(line, ">") )
    fieldKind = xeona::e_notField;

  // next complain if a field is not indented
  if ( fieldKind )                           // CAUTION: this test must be staged
    if ( ! std::isspace(line.at(0)) )        // test first char, requires <locale>
      {
        s_logger->repx(logga::warn, "non-indented field at line", d_lineCount);
      }

  // and finally ignore and discard everything after the
  // 'xeona::modelEndMarker' marker is encountered (note: could
  // later treat these lines as comments in a special record
  // kind, but that special record would need to be pushed back
  // as an afterthought)

  if ( d_flag_endMarker == true )
    {
      recordKind = xeona::e_notRecord;
      fieldKind  = xeona::e_notField;
    }

  // PROCESS A RECORD HEADER LINE

  if ( recordKind )
    {
      // create some variables
      std::string type;                      // record type (as string not enum)
      std::string identifier;                // record identifier

      // if appropriate, close the old record and begin a new record
      if ( d_flag_initialRecord == true )    // first time a record header has been seen
        {
          d_flag_initialRecord = false;      // toggle so future records can be closed
          s_logger->repx(logga::dbug,
                         "toggled initialRecord flag, now",
                         d_flag_initialRecord);
        }
      else
        {
          ++d_recCount;
          std::ostringstream put;
          put << "\n";                        // to cope with 'logga::xtra' reporting
          s_logger->putx(logga::xtra, put);
          put << "  pushing back curRecord " << d_recCount << "\n";
          s_logger->putx(logga::dbug, put);
          s_logger->addSmartBlank(logga::xtra);   // 'logga::xtra' is correct

          d_records.addRecord(d_curRecord);  // STL container elements are copied in
          d_curRecord.reset(new Record());   // reload smart pointer for reuse here
        }

      // note if special 'model-end' record header is encountered
      if ( recordKind == xeona::e_end )
        {
          s_logger->repx(logga::dbug,
                         "record-set end-marker encountered",
                         xeona::modelEndMarker);
          d_flag_endMarker = true;           // reset flag
          s_logger->repx(logga::dbug,
                         "toggled endMarker flag, now",
                         d_flag_endMarker);
        }

      split.clear();
      boost::split(split, line, boost::is_any_of("."));
```

```
      switch ( split.size() )
        {
        case 2:
          identifier = split[1];
          // CAUTION: no break because fall-thru is correct
        case 1:
          type       = split[0];
          break;
        default:
          identifier = "(not set)";
          type       = "(not set)";
          s_logger->repx(logga::warn, "record header not split into two", split.size());
        }
      boost::trim(type);
      boost::trim(identifier);

      // complain if record header lacks information
      if ( identifier.empty()
           && recordKind != xeona::e_note         // was && type != "note"
           && type != xeona::modelEndMarker )     // set to "model-end" at time of writing
        {
          s_logger->repx(logga::warn, "incomplete record header at line", d_lineCount);
          std::ostringstream put;
          put << "problem line : '" << oline << "'\n";
          s_logger->putx(logga::dbug, put);
          return false;
        }

      // load new information
      d_curRecord->addIdentifier(identifier);
      d_curRecord->addKind(recordKind);
      d_curRecord->addEnabled(lineEnabled);  // bool value
    }

  // PROCESS A FIELD LINE

  else if ( fieldKind )
    {
      // split into that part to left of '<' or '>' and that part
      // to right of same
      std::string left;                     // left side part
      std::string right;                    // right side part

      split.clear();
      boost::split(split, line, boost::is_any_of("<>"));    // multiple delimiters used
      if ( split.size() >= 3 )                              // should only be two parts
        s_logger->repx(logga::warn, "multi-way field split", split.size());
      left  = split[0];
      right = split[1];

      boost::trim(left);
      boost::trim(right);

      // undertake some integrity checks
      bool flag_problems = false;           // presume okay

      // complain if an enabled input line lacks a value
      if ( fieldKind == xeona::e_input
           && lineEnabled == true
           && right.empty() )
        {
          s_logger->repx(logga::warn, "input field lacks value at line", d_lineCount);
          flag_problems = true;
        }

      // complain if an enabled output line lacks a name
      else if ( fieldKind == xeona::e_output
                && lineEnabled == true
                && left.empty() )
        {
          s_logger->repx(logga::warn, "output field lacks name at line", d_lineCount);
          flag_problems = true;
        }

      // halt if required
      if ( flag_problems == true )
        {
          std::ostringstream put;
          put << "problem line : '" << oline << "'\n";
          s_logger->putx(logga::dbug, put);
          return false;
```

```
        }

      // PROCESS THE LEFT PART OF A FIELD LINE

      std::string name;
      std::string units;
      std::string remark;

      split.clear();
      boost::split(split, left, boost::is_any_of("[]"));    // multiple delimiters used
      switch ( split.size() )
        {
        case 1:                               // no units given, try to split again
          split.clear();
          boost::split(split, left, boost::is_any_of(" "), boost::token_compress_on);
          name   = split[0];
          boost::erase_first(left, name);
          remark = left;
          break;
        case 2:                               // units given, but no remark
          name   = split[0];
          units  = split[1];
          break;
        case 3:                               // units and remark given
          name   = split[0];
          units  = split[1];
          remark = split[2];
          break;
        default:
          s_logger->repx(logga::warn, "invalid split, size", split.size());
          break;
        }

      boost::trim(name);
      boost::trim(units);
      boost::trim(remark);

      // determine overall string length and bump 's_tabset' if necessary
      unsigned length = 0;
      if ( ! name.empty()   ) length += name.size()   + 1;
      if ( ! units.empty()  ) length += units.size()  + 1 + 2; // + 2 for the "[]"
      if ( ! remark.empty() ) length += remark.size() + 1;
      length += xeona::modelDisableChar.length()      + 1;     // for any "# "
      if ( length > s_tabset ) s_tabset = length; // then bump

      // load general information
      d_curField->addKind(fieldKind);
      d_curField->addEnabled(lineEnabled);

      // load new left part information
      d_curField->addName(name);
      d_curField->addUnits(units);
      d_curField->addRemark(remark);

      // SIMPLY LOAD THE RIGHT PART OF A FIELD LINE

       // load right part information as it stands
      d_curField->addRawStr(right);           // will be processed later in unit 'recset'

      // FINALIZE FIELD OBJECT

      d_curRecord->addField(d_curField);      // load current field
      d_curField.reset(new Field());          // reset current field
    }

  // PROCESS A COMMENT LINE

  else
    {
      if ( d_flag_initialRecord == false )   // a record header has been seen
        {
          // complain if earlier processed as a disabled line
          if ( lineEnabled == false )
            {
              std::string msg = "comment with leading '";
              msg += xeona::modelDisableChar;
              msg += "' at line" ;
              s_logger->repx(logga::warn, msg, d_lineCount);
            }
          d_curRecord->addComment(line);
        }
```

```
      else if ( ! line.empty() )             // meaning substantive pre-model material
        {
          s_logger->repx(logga::warn,
                         "pre-model material ignored",
                         boost::trim_left_copy(line));
          return false;
        }
    }

  // ADDITIONAL REPORTING

  std::string lineType;
  if      ( recordKind ) lineType = "record";
  else if ( fieldKind  ) lineType = "field";
  else                   lineType = "comment";

  std::ostringstream put;
  put << "  "
      << std::setw(8) << std::left  << lineType
      << std::setw(4) << std::right << d_lineCount
      << " : "
      << oline                               // could also use simply 'line'
      << "\n";
  s_logger->putx(logga::xtra, put);

  return true;                     // meaning no return false statements were encountered

} // DataIo::loadDataLine

// ------------------------------------------------------------
//  MEMBER FUNCTION : DataIo::locateModelHorizon
// ------------------------------------------------------------

unsigned
DataIo::locateModelHorizon()
{
  s_logger->repx(logga::xtra, "entering member function", "");

  unsigned steps = 0;                        // setting value

  const shared_ptr<Field> f
    = d_records.locateRecordAndField
    (xeona::timehorizon,                     // the "builtin." has been dropped,
     "steps");                               //   'xeona::timehorizon' is in 'common.cc'
  if ( ! f )
    {
      s_logger->repx(logga::warn, "horizon field not found, shared_ptr", f);
      return 0;
    }
  std::string rawHorizon = f->getRawStr();
  if ( rawHorizon.empty() )
    {
      s_logger->repx(logga::warn, "horizon string empty", "");
      return 0;
    }
  try
    {
      steps = boost::lexical_cast<unsigned>(rawHorizon);
    }
  catch( const boost::bad_lexical_cast& eblc )
    {
      s_logger->repx(logga::warn, "horizon string has bad lexical cast", rawHorizon);
      return 0;
    }
  Entity::setHorizonSteps(steps);
  return Entity::getHorizonSteps();

} // DataIo::locateModelHorizon

// ------------------------------------------------------------
//  MEMBER FUNCTION : DataIo::processModel
// ------------------------------------------------------------

void
DataIo::processModel()
{
  if ( d_records.getStatus() != RecordSet::e_loaded )  // confirm call order
    {
      std::ostringstream oss;
      oss << d_records.getStatus()  << " but wanting " << RecordSet::e_loaded;
      s_logger->repx(logga::warn, "data io call order error, status", oss.str());
```

```
      return;
    }
  d_records.processRecords();                          // process records
  d_records.setStatus(RecordSet::e_processed);         // update status
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::noteModelIsBound
// ----------------------------------------------------------

void
DataIo::noteModelIsBound()
{
  s_logger->repx(logga::dbug, "about to update d_records status", RecordSet::e_bound);
  d_records.setStatus(RecordSet::e_bound);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::updateModelRunTime
// ----------------------------------------------------------

void
DataIo::updateModelRunTime
(const std::string& currentSimRet,
 const std::string& currentSimKind)
{
  const std::string processid = boost::lexical_cast<std::string>(xeona::pid);
  const std::string svnrev    = boost::lexical_cast<std::string>(xeona::svnRev);

  updateProgramOutput("last-run", "process-id",      processid);
  updateProgramOutput("last-run", "used-svn",        svnrev);
  updateProgramOutput("last-run", "simulate-return", currentSimRet);
  updateProgramOutput("last-run", "run-kind",        currentSimKind);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::getSubset
// ----------------------------------------------------------

std::vector<shared_ptr<Record> >             // can be an empty vector
DataIo::getSubset                            // must come after 'processModel'
(const xeona::RecordKind recKind)            // subset based on record kind
{
  if ( d_records.getStatus() < RecordSet::e_processed )     // confirm call order
    {
      std::ostringstream oss;
      oss << d_records.getStatus()  << " but wanting " << RecordSet::e_processed;
      s_logger->repx(logga::warn, "data io call order error, status", oss.str());
      std::vector<shared_ptr<Record> > empty;
      return empty;
    }
  return d_records.copySubset(recKind);      // process request
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::writeModel
// ----------------------------------------------------------

void
DataIo::writeModel
(boost::filesystem::path modelFile)          // defaults to empty path
{
  s_logger->repx(logga::dbug, "entering member function", "");

  // CONFIRM CALL ORDER AND WRITE DEPTH

  switch ( d_records.getStatus() )
    {
    case RecordSet::e_bound:
      d_fieldWriteDepth = e_boundValue;
      s_logger->repx(logga::dbug, "proceeding with deep write", "");
      break;
    case RecordSet::e_processed:
      d_fieldWriteDepth = e_rawString;
      s_logger->repx(logga::dbug, "proceeding with shallow write", "");
      break;
    default:
      std::ostringstream oss;
      oss << d_records.getStatus()  << " but wanting " << RecordSet::e_bound;
      s_logger->repx(logga::warn, "data io call order error, status", oss.str());
      s_logger->repx(logga::warn, "abandoning model write", "");
```

```
      return;
    }

  if ( d_fieldWriteDepth == e_notSet )
    {
      s_logger->repx(logga::warn, "write depth not set", d_fieldWriteDepth);
    }

  // MAKE APPROPRIATE WRITE-OUT CALL

  if ( ! modelFile.empty() )
    {
      // process object name
      std::string filename  = modelFile.string();      // in operating system format
      const char* cfilename = filename.c_str();        // as C-string for fstream objects

      // open file in write mode and truncate mode, that is WITHOUT
      // specifying std::ios::app
      //
      // CAUTION: overwrite: the simple act of opening the file under
      // these conditions will cause its contents to be lost
      std::ofstream outfile(cfilename);        // defaults to text and std::ios::out
      if ( ! outfile )
        {
          std::cout << std::flush;
          std::clog
            << "\n"
            << "** unsuccessful ofstream construction on: " << cfilename << "\n"
            << "\n"
            << std::endl;
        }
      else
        {
          s_logger->repx(logga::dbug, "about to overwrite file", modelFile.filename());
          writeData(outfile);
        }
    } // on block exit, 'outfile' closes
  else
    {
      // write to console instead, most probably for testing purposes
      s_logger->repx(logga::dbug, "no path so will write to console", "stdlog");
      writeData(std::clog);
    }

  // UPDATE STATUS

  d_records.setStatus(RecordSet::e_written); // update status
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::setFieldWriteDepth
// ----------------------------------------------------------
//  Description  : modifies the field write depth
//  Role         : test purposes
// ----------------------------------------------------------

void
DataIo::setFieldWriteDepth
(FieldWriteDepth fwd)
{
#ifndef _XUTEST
  s_logger->repx(logga::kill, "call intended for unit testing only", "");
#endif // _XUTEST

  FieldWriteDepth prior = d_fieldWriteDepth;
  d_fieldWriteDepth     = fwd;

  std::ostringstream oss;
  oss << prior << " > " << d_fieldWriteDepth;
  s_logger->repx(logga::dbug, "field write depth reset from > to", oss.str());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::updateProgramOutput
// ----------------------------------------------------------
//  Description  : updates various program fields
// ----------------------------------------------------------

bool
DataIo::updateProgramOutput
(const std::string& recordName,              // sought record name
```

```
 const std::string& fieldName,               // sought field name
 const std::string& overwriteStr)            // overwrite if field exists and is output
{
  const shared_ptr<Field> f = d_records.locateRecordAndField(recordName, fieldName);
  if ( ! f )
    {
      std::ostringstream oss;
      oss << recordName << " " << fieldName;
      s_logger->repx(logga::dbug, "unable to find field", oss.str());
      return false;
    }
  const xeona::FieldKind kind = f->getKind();
  if ( kind != xeona::e_output )
    {
      std::ostringstream oss;
      oss << xeona::e_output << " | " << kind;
      s_logger->repx(logga::warn, "wrong field kind, wanted | got", oss.str());
      return false;
    }
  f->addRawStr(overwriteStr);
  return true;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::writeData
// ----------------------------------------------------------

void
DataIo::writeData                             // used by 'writeModel'
(std::ostream& os)
{
  s_logger->repx(logga::dbug, "entering member function", "");
  s_logger->repx(logga::dbug, "about to write out model", "");
  s_logger->flush();

  bool early = true;                          // used to control logging

  enum LastField                              // controls input and output field blanklines
    {
      e_unknown = 0,
      e_input   = 1,
      e_output  = 2
    };

  const FieldWriteDepth fwd = d_fieldWriteDepth;  // record for future reference

  const std::string disableChar = xeona::modelDisableChar;
  const unsigned fieldIndent    = xeona::modelFieldIndent;

  os << "\n";                                 // leading blank line

  // cycle thru records: const std::vector<shared_ptr<Record> >&
  BOOST_FOREACH( shared_ptr<Record> r, d_records.getRecords() )
    {
      // first pass logging
      if ( early ) s_logger->repx(logga::adhc, "processing early record at", r.get());

      // useful check
      if ( ! r ) s_logger->repx(logga::warn, "record pointer is null or empty", r);

      // grab values
      xeona::RecordKind recKind = r->getKind();
      bool recEnabled           = r->getEnabled();
      std::string recIdentifier = r->getIdentifier();

      // useful reporting
      s_logger->repx(logga::xtra, "record name", recIdentifier);

      // process header line (the entity identifier or similar)
      std::string header;
      if ( recEnabled == false )       header += disableChar + " ";
      header                                  += recordKindToLead(recKind);     // [1]
      if ( ! recIdentifier.empty() )   header += "." + recIdentifier;

      // [1] might be better implemented as a dictionary

      os << header;
      os << "\n";                             // final newline

      // set shallow for non-entities and disabled entities
      if ( recKind == xeona::e_program
```

```
     ||
     recEnabled == false )
  d_fieldWriteDepth = e_rawString;     // opt for shallow write
else
  d_fieldWriteDepth = fwd;             // revert to stored value

bool flag1 = true;                     // controls blank line squeezing in comments
bool flag2 = true;                     // controls printing of final blank line

// cycle thru fields: const std::vector<shared_ptr<Field> >&
LastField lastField = e_unknown;
bool addBlank       = true;            // controls blanks between input and output
BOOST_FOREACH( shared_ptr<Field> f, r->getFields() )
  {
    // first pass logging
    if ( early ) s_logger->repx(logga::adhc, "processing first field at", f.get());

    // useful check
    if ( ! f ) s_logger->repx(logga::warn, "field pointer is null or empty", r);

    // grab values
    std::string      fiName    = f->getName();
    xeona::FieldKind fiKind    = f->getKind();
    tribool          fiEnabled = f->getEnabled();
    std::string      fiUnits   = f->getUnits();
    std::string      fiRemark  = f->getRemark();
    std::string      fiRawStr  = f->getRawStr();

    // process field line left
    std::string left;
    if ( fiEnabled == false )    left += disableChar + " ";
    left                              += fiName;
    if ( ! fiUnits.empty() )     left += " [" + fiUnits + "]";
    if ( ! fiRemark.empty() )    left += " "  + fiRemark;

    // process field line right
    std::string right;

    if ( fiName == "class" )      // meaning class name rather than entity value
      {
        right                         += " >";
        right                         += getFieldValue(f, e_rawString);
      }
    else if ( fiKind == xeona::e_input )
      {
        right                         += " >";
        if ( fiEnabled )
          right                       += getFieldValue(f);      // important call
        else                                                    // shallow recovery
          right                       += getFieldValue(f, e_rawString);
        if ( lastField == e_output ) addBlank = true;
        lastField = e_input;
      }
    else if ( fiKind == xeona::e_output )
      {
        right                         += " <";
        if ( fiEnabled )
          right                       += getFieldValue(f);      // important call
        else                                                    // shallow recovery
          right                       += getFieldValue(f, e_rawString);
        if ( lastField == e_input ) addBlank = true;
        lastField = e_output;
      }
    else
      {
        s_logger->repx(logga::warn, "field kind not class, input, output", fiKind);
      }

    // add blank and update state toggle
    if ( addBlank )
      {
        addBlank = false;
        os << "\n";
      }

    // reduce indent for disabled fields in order to
    // improve alignment (some of the emacs xeona editing
    // major modes, including 'xem-mode', expect this)
    unsigned indent = fieldIndent;
    unsigned tabset = s_tabset;
    const int shift = 2;               // length of "# " string
```

```
          if ( fiEnabled == false && indent >= 4 )
            {
              indent = indent - shift;
              tabset = tabset + shift;
            }

          // write out current field
          os << boost::format("%1%") % boost::io::group(std::setw(indent), "")
             << boost::format("%1%") % boost::io::group(std::left, std::setw(tabset),left)
             << boost::format("%1%") % right
             << "\n";                                  // final newline

          if ( fiName == "class" ) addBlank = true;

          flag2 = true;
          early = false;                      // reset logging flag

        } // FOREACH shared_ptr<Field>

      // cycle thru multi-line comments: const std::vector<std::string>&
      BOOST_FOREACH( std::string s, r->getComments() )
        {
          if ( s.empty() && flag1 )
            {
              os << s << "\n";
              flag1 = false;
              flag2 = false;                  // used shortly
            }
          else if ( ! s.empty() )
            {
              // modify "tab-stop-list"
              modifyEmacsTabstops(s);         // added after main code tested
              os << s << "\n";
              flag1 = true;
            }
        } // FOREACH std::string

      if ( flag2 )                            // set in comment loop about 60 lines above
        {
          os << "\n";                         // end-of-record blank line as required
        }

    } // FOREACH shared_ptr<Record>

  os << xeona::modelEndMarker;
  os << "\n";                                 // end-of-file blank line
  os << std::endl;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : DataIo::recordKindToLead
// ------------------------------------------------------------

// must align with code in 'DataIo::loadDataLine'

std::string
DataIo::recordKindToLead
(xeona::RecordKind kind) const
{
  std::string lead;
  switch ( kind )
    {
    case xeona::e_note:    lead = "note";    break;
    case xeona::e_program: lead = "program"; break;
    case xeona::e_entity:  lead = "entity";  break;
    case xeona::e_end:     lead = "end";     break;
    default: std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
    }
  return lead;
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : DataIo::getFieldValue
// ------------------------------------------------------------

std::string                                   // formatted for writing out
DataIo::getFieldValue
(const shared_ptr<Field> f,
 FieldWriteDepth         fwd)                 // default 'e_notSet' given
{
  const std::string sep = " ";                // locally hardcoded timeseries separator
```

```
  std::string buf;                          // holds the output

  // additional reporting as appropriate
  // YEEK 33 CODE (set by '--yeek')
  //
  // prior to commit r4814, empty output automatically reported
  // "(some problem)" -- this however prevented a possibly
  // incomplete model from being rerun without prior cleaning
  // (for instance using something like: sed 's/(some problem)//')
  //
  // this code makes "" the default and allows the previous
  // behavior available via --yeek 33

  if ( xeona::yeek == 33 )
    {
      buf = xeona::modelStringDelim
        + "(some problem)"
        + xeona::modelStringDelim;

      const std::string name = f->getName();
      s_logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
      s_logger->repx(logga::warn, "added (some problem) to output", name);
    }
  else
    {
      buf = xeona::modelStringDelim
        + xeona::modelStringDelim;
    }

  if ( fwd == e_notSet )                    // most likely no second argument given
    {
      fwd = d_fieldWriteDepth;              // thus opt for the current recorded state
    }
  switch ( fwd )
    {
    case e_rawString:                       // shallow write for test purposes only
      {
        buf = f->getRawStr();               // simple grab raw string
      }
      break;

    case e_boundValue:                      // deep write
      {                                     // CAUTION: a local block is necessary [1]
        const int length = f->getCount();
        s_logger->repx(logga::xtra, "field name", f->getName());
        s_logger->repx(logga::xtra, "field length", length);

        // DESIGN CONSIDERATION: as coded, the only thing that
        // distinguishes singles from timeseries is length,
        // meaning that timeseries have a element count value
        // greater than unity.  However, if some other attribute
        // was used (say a trailing + on the field name) then
        // timeseries of length unity could be used and a horizon
        // steps value of one would be valid.  Development
        // testing confirms that next switch block is the ONLY
        // place that code changes would need to be made.

        switch ( length )
          {
          case 0:
            {                               // CAUTION: a local block is necessary [1]
              std::ostringstream oss;
              oss << "details above if report " << logga::xtra;
              s_logger->repx(logga::rankNoData, "bound value is empty", oss.str());
            }
            break;
          case 1:
            {
              buf = f->getSingle();         // const std::string
              s_logger->repx(logga::xtra, "single as string", buf);
            }
            break;
          default:                          // longer
            {
              buf = f->getTimeseries(sep);  // const std::string
              s_logger->repx(logga::xtra, "timeseries as string", buf);
            }
            break;
          } // inner switch statement
      }
      break;
```

```
      // [1] else error messages like (which puzzle me):
      // c/datio.cc:1072: error: jump to case label
      // c/datio.cc:1057: error:   crosses initialization of 'const int length'

    default:
      {
        s_logger->repx(logga::warn, "coding error, d_fieldWriteDepth", d_fieldWriteDepth);
      }
      break;
    } // outer switch statement

  if ( ! buf.empty() )
    {
      buf = " " + buf;                        // append a leading space
    }

  return buf;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : DataIo::modifyEmacsTabstops
// ----------------------------------------------------------
//  Description  : updates 'tab-stop-list' value if present
//  Role         : used on model comments write to update 'emacs' local variables settings
//  Techniques   : Boost.String_algo library
//  Status       : complete
//
//  Design notes
//
//      typical entry: '    tab-stop-list: (04 45 47)'
//
//  CAUTION: late addition
//
//      Be aware that 'modifyEmacsTabstops' was added some months
//      after the original coding and testing of this unit.
//
// ----------------------------------------------------------

void
DataIo::modifyEmacsTabstops
(std::string& commentLine) const
{
  // define trigger phrase
  const std::string emacsSymbol = "tab-stop-list";

  // preamble
  std::ostringstream put;

  // look for trigger phrase
  if ( boost::contains(commentLine, emacsSymbol) )
    {
      std::string buffer(commentLine);       // to be copied later to 'commentLine'
      put << "  emacs symbol found : "  << emacsSymbol   << "\n"
          << "    original line    : '" << buffer << "'" << "\n";
      s_logger->putx(logga::dbug, put);

      // calculations as per '~DataIo' in this file
      int tab1 = xeona::modelFieldIndent;    // set in 'common.cc'
      int tab2 = tab1 + s_tabset + 1;        // determined here
      int tab3 = tab2 + 2;

      // format new tab-stop-list
      std::ostringstream ssBuffer;
      ssBuffer << "(" << boost::format("%02d") % tab1
               << " " << boost::format("%02d") % tab2
               << " " << boost::format("%02d") % tab3
               << ")";
      std::string tabstoplist = ssBuffer.str();

      // modify existing line
      const std::string listOpen = "(";      // denotes beginning of emacs list
      std::string::size_type index = buffer.find(listOpen);
      if ( index != std::string::npos )
        {
          buffer.erase(index, buffer.length());   // chop
          buffer += tabstoplist;                  // append
          put << "    replacement line : '" << buffer << "'" << "\n";
          s_logger->putx(logga::dbug, put);
          commentLine = buffer;              // overwrite function input
        }
```

```
      else
        {
          s_logger->repx(logga::info,
                         "emacs " + emacsSymbol + " lacks \"(\"",
                         "");
        }
    }
  else
    {
#ifdef _XUTEST                                  // unit test reporting
      std::ostringstream put;
      put << "  emacs symbol NOT found : " << emacsSymbol << "\n";
      s_logger->putx(logga::dbug, put);
#endif // _XUTEST
    }
}

//  $Source: /home/robbie/synk/xeona/fragments/RCS/frag-boost-lexical-1.cc,v $
//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : extunits.cc
//  file-create-date : Wed 02-Dec-2009 17:24 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : quantifying extensity enums and interpretation / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/extunits.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "extunits.h"         // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

namespace xeona
{
  std::string
  interpretExtensity
  (const xeona::ExtensityUnit extensity)
  {
    std::string buf = "(not overwritten)";
    switch ( extensity )
      {
      case xeona::notSpecified: buf = "(not specified)"; break;
      case xeona::joule:        buf = "J";               break;
      case xeona::kilogram:     buf = "kg";              break;
      case xeona::uoa:          buf = "$";               break;
      case xeona::metreSq:      buf = "m2";              break;
      default: std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
      }
    return buf;
  }

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : factory.cc
//  file-create-date : Tue 22-May-2007 12:59 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : entity sub-class factory / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/factory.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "factory.h"          // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../a/logger.h"      // run-time logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting

//  CODE

// ----------------------------------------------------------
//  CLASS           : EntityFactory
// ----------------------------------------------------------
//  Description  : an enum-keyed entity object factory
//  Patterns     : concrete object factory, singleton
//  Status       : complete
//
//  Design notes
//
//      This class used two classic design patterns: the concrete
//      object factory and the singleton.  Code for both drew
//      heavily on Alexandrescu (2001).  Somewhat similar
//      material is presented in Stephens etal (2006) but is not
//      discussed as comprehensively.
//
//      The singleton is a build-on-first-request object, which,
//      in this case, lasts until some time after the end of
//      'main()'.  The static 's_instance' pointer relies on
//      "static initialization" meaning it is a type without
//      constructor initialized with a compile-time constant.
//      This strategy is, for reasons not give here, inherently
//      safe.
//
//      The object factory was originally coded with const
//      std::string type identifiers, but these were changed to
//      xeona::EntityType enums to enable compile-time checking.
//      Then r2209 reverted to the original approach.
//
//      The use of a smart (rather than raw) pointer for
```

```
//      's_instance' prevented a small memory leak.
//
//      The factory pattern will be required when
//      "deserialization" is introduced.  Deserialization is the
//      process of creating C++ objects from information stored
//      on disk.  Further thought will be required to implement
//      this functionality.
//
//      Finally, apologies to Alexandrescu for not taking his
//      advice and hand crafting this class to a much larger
//      degree than he might consider sensible.
//
//  Additional comment regarding singletons
//
//      With hindsight, it is not worth implementing formal
//      singletons.  It is quite sufficient to add a constructor
//      counter and log warnings on second and subsequent calls.
//
//  CAUTION: interacting singletons
//
//      Coordinating the life expectancy, and more specifically,
//      the destruction order of interacting singleton objects is
//      both subtle and difficult -- under the current code (and
//      build environment), the 's_logger' Logger object may be
//      destroyed before the 'EntityFactory' singleton is
//      destroyed.  Hence, 's_logger' calls placed in
//      ~EntityFactory() may result in unpredictable behavior.
//
//      Alexandrescu (2001 pp126-156) provides an excellent
//      description of the issues involved.  Read his chapter on
//      singletons if you need to know more.
//
//  References
//
//      Alexandrescu, Andrei.  2001.  Modern C++ design : generic
//          programming and design patterns applied.  Addison-Wesley,
//          Boston, USA.  ISBN 0-201-70431-5.
//
//      Stephens, D Ryan, Christopher Diggins, Jonathan Turkanis,
//          and Jeff Cogswell.  2006.  C++ cookbook : solutions
//          and examples for C++ programmers.  O'Reilly Media,
//          Sebastopol, California, USA.  ISBN 0-596-00761-2.
//
// ------------------------------------------------------------

//  STATIC DEFINITIONS

EntityFactory::entityFactory_type
EntityFactory::s_instance = EntityFactory::entityFactory_type::shared_ptr(); // [1]

// [1] creates an empty smart pointer (the following will also
// work = EntityFactory::entityFactory_type();) -- for more
// details on empty, null, single, and copied shared_ptr, see
// documentation for the free function 'xeona::reportShared_ptr'.

logga::spLogger
EntityFactory::s_logger = logga::ptrLogStream();  // bind logger on definition

// CREATORS

EntityFactory::EntityFactory() :             // CAUTION: private
  d_callbacks()
{
  s_logger->repx(logga::dbug, "constructor call", "singleton pattern");
}

EntityFactory::~EntityFactory()
{
  // CAUTION: 'Logger' object calls placed here will cause
  // trouble if the 'Logger' singleton has already been destroyed
  // -- at the time of writing, this is not currently a problem
  // but there are NO GUARANTEES (see the comments in factory
  // unit test for more information)

  // 0 = do nothing
  // 1 = call with Logger object
  // 2 = call without (not under logga::Rank control)

#if !defined(XE_FACTORY_DTOR_LOGGING)
# define XE_FACTORY_DTOR_LOGGING 0
#endif
```

```
#if   (XE_FACTORY_DTOR_LOGGING == 0)
  // do nothing
#elif (XE_FACTORY_DTOR_LOGGING == 1)
  s_logger->repx(logga::dbug, "destructor call", "placement okay?");
#elif (XE_FACTORY_DTOR_LOGGING == 2)
  xeona::logDirect(__FILE__, __LINE__, __func__, "destructor call", "logDirect report");
#endif // XE_FACTORY_DTOR_LOGGING
}

// SINGLE POINT OF ACCESS

// 'instance' could return an 'EntityFactory&' reference, but
// here 'instance' provides a dereferenced *s_instance -- and
// therefore indirection is required in the client code:
//
//     EntityFactory::instance()->someFn()

EntityFactory::entityFactory_type
EntityFactory::iface()
{
  if ( ! s_instance )                       // implicit boolean conversion
    {
      entityFactory_type smartptr(new EntityFactory());     // constructor call
      s_instance = smartptr;                                // store locally
      s_logger->repx(logga::dbug, "first time, s_instance ptr", s_instance);
    }
  else
    {
      s_logger->repx(logga::adhc, "subsequent, s_instance ptr", s_instance);
    }
  return s_instance;
}

// MANIPULATORS

bool                                        // true if successful
EntityFactory::registerEntity
(const std::string    entityRegn,
 createEntityCallback createfn)
{
  s_logger->repx(logga::xtra, "about to register entity type", entityRegn);

  // for sets and maps, insert() returns a std::pair<iterator,
  // bool> signalling position and success, and here 'second'
  // refers to this struct and NOT our make_pair struct (Josuttis
  // 1999 p183, Alexandrescu 2001 p205)

  return d_callbacks.insert(std::make_pair(entityRegn, createfn)).second;
}

bool                                        // true if previously registered
EntityFactory::unregisterEntity
(const std::string entityRegn)
{
  s_logger->repx(logga::dbug, "about to UNregister entity type", entityRegn);

  // member function 'erase()' returns number of elements removed
  return d_callbacks.erase(entityRegn) == 1;
}

shared_ptr<Entity>
EntityFactory::createEntityBind
(const std::string  entityRegn,
 const std::string& entityId,
 Record&            r)                     // live record
  throw(std::exception,                     // exception specification
        xeona::empty_wrap,
        xeona::non_registration)
{
  s_logger->repx(logga::xtra, "processing entity", entityId);
  callback_map::const_iterator pos;
  pos = d_callbacks.find(entityRegn);
  if ( pos == d_callbacks.end() )
    {
      s_logger->repx(logga::warn, "entity registration not found", entityRegn);

      // if '--krazy' return empty pointer and continue,
      // otherwise notify and exit

      if ( xeona::nopro )
```

```
        {
          std::ostringstream put;
          put << "** unregistered entity class requested"                          << "\n"
              << "       using '--krazy' code"                                     << "\n"
              << "       entity identifier   : " << entityId                       << "\n"
# ifndef _XUTEST // not unit testing [1]
              << "       requested class     : " << r.locateClass()                << "\n"
# endif // _XUTEST
              << "       entity registration : " << entityRegn                     << "\n"
              << "   about to return empty pointer of type 'shared_ptr<Entity>'"   << "\n"
              << "   this will almost certainly cause more problems downstream"    << "\n";
          s_logger->putx(logga::warn, put);

          return shared_ptr<Entity>::shared_ptr();    // empty pointer
        }
      else
        {
# ifndef _XUTEST // not unit testing [1]
          const std::string requestedClass = r.locateClass();
# else
          const std::string requestedClass = "(not available for unit test)";
# endif // _XUTEST
          s_logger->repx(logga::dbug, "will throw xeona::non_registration", "");
          throw xeona::non_registration(entityId, entityRegn, requestedClass);
        } // xeona::nopro

      // [1] gave rather puzzling link-time error for unit test:
      //     b/entity.o: In function `__tcf_13':
      //     collect2: ld returned 1 exit status

    } // if ( pos == d_callbacks.end() )

  // note the following public typedef declared within class
  // 'EntityFactory':
  //
  //   shared_ptr<Entity> (*createEntityCallback)(const std::string&, Record&)

  return (pos->second)(entityId, r);         // invoke creation function
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : files.cc
//  file-create-date : Fri 09-Nov-2007 13:38 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions for regular files / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/files.cc $

//  AD-HOC NOTES
//
//  Boost.Filesystem
//
//      Here are some Boost.Filesystem portability issues which
//      were not covered well in the Boost documentation at the
//      time of coding.
//
//  Path stringification functions
//
//      string()
//
//          returns a POSIX compliant string with no distinction
//          between regular files and directories -- and (for
//          most normal characters, including " " and "\\") no
//          different from native Linux
//
//      file_string()
//
//          returns an operating system format string for regular
//          files
//
//      directory_string()
//
//          returns an operating system format string for
//          directories -- noticeable only on systems (VMS but
//          not Linux or Windows) for which the formatting rules
//          for regular files and directories differ
//
//      stream insertion via operator<<
//
//          stream insertion mimics string()
//
//      native_*_string()
//
//          depreciated
//
//  Path completion functions
//
//      complete()
//
//          assumes POSIX compliant input
//
//      system_complete()
```

```
//
//          assumes operating system format input
//
//  Portability conclusions
//
//      input : all input assumes POSIX path conventions
//
//          complete() is used, even where user input applies,
//          which implies user input should be POSIX compliant
//
//      reporting : all reporting uses OS path conventions
//
//          file_string() is used to stringify paths for
//          reporting (unless local circumstances dictate
//          otherwise)
//
// -----------------------------------------------------

//  LOCAL AND SYSTEM INCLUDES

#include "files.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <fstream>            // file-based io
#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <cerrno>             // C-style error codes, errno object

#include <sys/stat.h>         // C-style POSIX file characteristics, stat()
#include <sys/types.h>        // C-style POSIX primitive system data types

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/any.hpp>                // type heterogeneous storage
#include <boost/cast.hpp>               // numeric_cast<> number to number conversions
#include <boost/format.hpp>             // printf style formatting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

//  CODE

namespace xeona
{
  // -----------------------------------------------------------
  //  FREE FUNCTION   : xeona::testWritable
  // -----------------------------------------------------------
  //  Description  : check if a given Boost.Filesystem path is writable
  //  Techniques   : relies exclusively on C++ Standard Library calls
  //  Status       : complete
  //
  //  Design notes
  //
  //      This function uses only C++ Standard Library calls from
  //      <fstream>.  It does not, for instance, rely on:
  //
  //          * the C run-time function stat() from <sys/stat.h>,
  //            which not all compilers supply
  //
  //          * the Boost.Filesystem library, which currently has
  //            no "boost::filesystem::is_writable()" function anyway
  //
  //          * the Unix-specific access() function from <unistd.h>
  //
  //          * OS-specific support from Visual C++ and other
  //            dedicated Windows compilers
  //
  //      Lischner (2003), as always, is good on the specifics of
  //      Standard Library headers.
  //
  //  References
  //
  //      Lischner, Ray.  2003.  C++ in a nutshell : a language and
  //        library reference, O'Reilly and Associates, Sebastopol,
  //        California, USA.  ISBN 0-596-00298-X.
  //
  // -----------------------------------------------------------
```

```
tribool                                 // indeterminate means status not resolved
testWritable
(boost::filesystem::path regularFile)   // this function assumes the file is regular
{
  // bind logger object
  // static logga::spLogger logger = logga::ptrLogStream();

  // recover path name
  std::string pathname  = regularFile.string();      // use native format
  const char* cpathname = pathname.c_str();          // C-style string

  // some subtle logic is required for the writability test
  // because first the 'ifstream' call and then the 'ofstream'
  // call must succeed -- the 'ifstream' test is required
  // because it fails because the file in question is absent,
  // the 'ofstream' test will simply create the missing file
  // and return success (not what is wanted)

  tribool writable = false;             // set default return variable
  std::ifstream i;
  i.open(cpathname, std::ios_base::binary | std::ios_base::in);
  if ( i )                              // read-only binary open succeeded
    {
      i.close();
      std::ofstream o;
      o.open(cpathname, std::ios::binary | std::ios::out | std::ios::app); // [1]
      if ( o )                          // write-only binary append open succeeded
        {
          o.close();
          writable = true;
        }
    }

  // CAUTION: [1] file open mode: ofstream "app" append mode is
  // essential, otherwise the default "trunc" truncate mode
  // will nuke the existing file

  return writable;
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::establishModelFile
// ----------------------------------------------------------
//  Description  : build path and confirm model file or show failure
//  Takes        : the model name or alternatively an empty string
//  Status       : complete
//
//  Design notes
//
//      If passed an empty string, this function uses the
//      built-in default name (model) from 'common.cc'
//
//      Takes this name and concatenates the built-in extension
//      (.xem) from 'common.cc'.  And then converts this to a
//      Boost.Filesystem path and, in the process, adds the
//      initial directory from which the application was invoked.
//
//      Checks the path is a regular file.  Then calls
//      'testWritable()' to confirm write status.  And finally
//      backs up the file using the backup tag (~) set in
//      'common.cc'.
//
//      Returns an empty path on failure, otherwise the actual
//      path.  An empty path tests true with path.empty().
//
//      Regarding terminology, a "path" can be a regular file or
//      a directory.  A "leaf" is the path with the parent
//      directory and such omitted.  A file "extension" contains
//      the dot as well, thereby allowing a trailing dot and no
//      extension to be distinguished.  A "complete" path means
//      an absolute path.
//
//  CAUTION: file path format conventions used in this code
//
//      input     : all input assumes POSIX path conventions
//      reporting : all reporting uses OS path conventions
//
//  See also
//
//      Notes on Boost.Filesystem elsewhere in this file.
//
```

```
//  References
//
//      Boost.Filesystem documentation.
//
// ------------------------------------------------------------

boost::filesystem::path                     // an empty path object indicates failure
establishModelFile                          // build path and confirm model file
(const std::string& modelStub)              // from command-line or "" to use default
{
  static logga::spLogger logger = logga::ptrLogStream();
  logger->repx(logga::dbug, "entering free function", "");

  // CAUTION: advice on the hardcoding of paths (including
  // incomplete paths): the solution adopted here is to use legal
  // Linux pathnames and avoid the use of escaped characters --
  // these paths (and subpaths), then, will be automatically
  // POSIX conforming.

  // adopt a simplified namespace alias for convenience
  namespace fs = boost::filesystem;

  // create empty return object in the event of failure, thus empty.empty() returns true
  fs::path empty;        // default (zero-argument) construction creates an empty object

  // use the default model name from 'common.cc' if an empty string was passed
  std::string modelName;
  if ( modelStub.empty() )                  // 'modelStub' is a function argument
    modelName = xeona::modelStubDefault;    // set in 'common.cc'
  else
    modelName = modelStub;

  // form the leaf names for the model file and the backup file
  std::string modelLeaf;
  std::string backupLeaf;
  modelLeaf   = modelName;                  // say : subdir/model2
  modelLeaf  += xeona::modelExt;            // thus: subdir/model2.xem
  backupLeaf  = modelLeaf;
  backupLeaf += xeona::backupTag;           // thus: subdir/model2.xem~

  // form complete (absolute) paths for the model file and the backup file
  fs::path model  = fs::absolute(modelLeaf);   // note the default second argument [1]
  fs::path backup = fs::absolute(backupLeaf);

  // [1] boost::filesystem::absolute() has a default second
  // argument of boost::filesystem::initial_path(), this being
  // the current directory at the time of entry into main().
  //
  // CAUTION: system_complete has different behavior so read the
  // documentation if you wish to change from POSIX conformance
  // to operating system-based format rules

  // regenerate the leaf names in native format for use in local reporting
  std::string natModelLeaf;
  std::string natBackupLeaf;
  fs::path temp;
  temp          = model.filename();
  natModelLeaf  = temp.string();
  temp          = backup.filename();
  natBackupLeaf = temp.string();

  // check for the presence of the model file
  if ( ! fs::is_regular_file(model) )       // path exists and is a regular file [2]
    {
      logger->repx(logga::warn, "model file not found", natModelLeaf);
      return empty;
    }

  // [2] boost::filesystem::is_regular() is similar to the bash
  // builtin: test -f FILE and boost::filesystem::exists() is
  // similar to: test -e FILE (in this case FILE includes regular
  // files, directories, symlinks, and specials)

  // confirm file is writable with C++ <fstream> calls -- while
  // noting that there is currently no such support in the
  // Boost.Filesystem library

  if ( xeona::testWritable(model) != true )     // free function from this file
    {
      logger->repx(logga::warn, "model file not proved writable", "");
      return empty;
```

```
    }

  // remove any existing backup file -- an existence test is not
  // necessary because non-existence does not create an exception
  // although the call itself will return 'false'

  try
    {
      fs::remove(backup);
    }
  catch( const std::exception& e )
    {
      logger->repx(logga::warn, "boost::filesystem::remove() fail", natBackupLeaf);
      std::clog << std::flush;
      std::cout
        << "\n"
        <<"** boost::filesystem::remove(" << backup.string() << "): "
        << e.what()
        << "\n"
        << std::endl;
      return empty;
    }

  // back-up the current model file

  try
    {
      fs::copy_file(model, backup);        // CAUTION: 'backup' MUST NOT exist
      logger->repx(logga::info, "model file now backed up", "");
    }
  catch( const std::exception& e )
    {
      logger->repx(logga::warn, "boost::filesystem::copy_file fail", "");
      std::clog << std::flush;
      std::cout
        << "\n"
        <<"** boost::filesystem::copy_file("
        << model.string()  << ", "
        << backup.string() << "): "
        << e.what()
        << "\n"
        << std::endl;
      return empty;
    }

  // additional reporting, CAUTION: leave as std::cout because
  // this is to be treated as application reporting and not
  // logging

  logger->addSmartBlank();                // next logger call should add blank line
  std::cout
    << ""                                                    << "\n"
    << "  model details (backup made)"                       << "\n"
    << ""                                                    << "\n"
    << "     model file        :  " << model.string()        << "\n"
    << "     backup file       :  " << backup.string()       << "\n"
    << std::flush;

  // make a substantive (rather than empty) path return
  logger->repx(logga::dbug, "returning a useful model path", natModelLeaf);
  return model;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::readonly
// ------------------------------------------------------------
//  Description  : chmod given 'filename' to 'newMode' (see implementation)
//  Role         : general use (but often after various GLPK file creation calls)
//  Headers      : <sys/stat.h> <sys/types>
//  Techniques   : POSIX 'chmod'
//  Status       : complete
//
//  Design notes
//
//      The 'chmod' call conforms to POSIX.
//
//      See primarily $ man 2 chmod.  See also Robbins and
//      Robbins (2003 p105) for a list of symbolic file
//      permission names and 'errno' codes.
//
//  References
```

```
  //
  //      Robbins, Kay A and Steven Robbins.  2003 UNIX systems
  //        programming : communication, concurrency, and threads --
  //        Second edition.  Prentice Hall PTR, Upper Saddle River,
  //        New Jersey, USA.  ISBN 0-13-042411-0.
  //
  // ----------------------------------------------------------

  bool                                      // 'false' if unsuccessful
  readonly
  (const char* filename)
  {
    if ( filename == NULL )                 // protect against NULL char*
      {
        return readonly("");
      }
    else
      {
        const std::string buf(filename);    // C-string conversion
        return readonly(buf);               // simple wrapper to the std::string variant
      }
  }

  bool                                      // 'false' if unsuccessful
  readonly
  (const std::string& filename)
  {
    //  CAUTION: the POSIX symbolic names S_IRGRP and S_IROTH are
    //  not defined in Windows as there is no way to deal with
    //  file permissions for groups and others.  The following
    //  code protects against undefined macros.  (This fix from a
    //  posting by Aleksander Morgado on 17-Apr-2008, recovered
    //  on 16-Mar-2010.  See also Robbins and Robbins (2003 p105)
    //  for a complete list of POSIX file permissions.)

    // define new mode using POSIX symbolic file permission names
    mode_t newMode = S_IRUSR;               // UNIX u=r or Windows read-only
#ifdef S_IRGRP
    newMode |= S_IRGRP;                     // add UNIX g=r
#endif

    // bind logger and prepare 'modestr'
    static logga::spLogger logger = logga::ptrLogStream();
    std::ostringstream oss;
    oss << std::setw(4) << std::setfill('0') << std::oct << newMode;
    const std::string modestr = oss.str();

    // active code
    logger->repx(logga::adhc, "chmod file to " + modestr, filename);
    if ( chmod(filename.c_str(), newMode ) == -1 )     // POSIX system call
      {
        // call failure
        std::ostringstream put;
        put << "  xeona::readonly call failed"  << "\n"
            << "    filename    : " << filename << "\n"
            << "    new mode    : " << modestr  << "\n"
            << "    POSIX call  : " << "chmod"  << "\n"
            << "    POSIX errno : " << errno    << "\n";  // refer <cerrno>
        logger->repx(logga::warn, "chmod call failed, details follow", "");
        logger->putx(logga::warn, put);
        logger->addSmartBlank(logga::warn);
        return false;
      }
    return true;
  }

} // namespace xeona

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : fincalc.cc
//  file-create-date : Fri 17-Oct-2008 14:59 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : support for discounted cash flow analysis / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/fincalc.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Note
//
//      In particular, see the test comparisons at the end.
//
//  Source of financial equations
//
//      wikipedia page title: Time value of money
//      retrieved from: http://en.wikipedia.org/wiki/Time_value_of_money
//      page last modified : 26 March 2008 at 19:36
//
//  See also
//
//      xeona design notes on costs and prices
//      currently file: xeona-commentary_11.odt

//  LOCAL AND SYSTEM INCLUDES

#include "fincalc.h"          // companion header for this file (place first)

#ifndef _XUTEST
#include "../b/entity.h"      // entity base class plus lazy linking
#endif // XE_XUTEST

#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <cmath>              // C-style maths, ceil(), floor(), sqrt()
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <iostream>           // standard io
#include <iomanip>            // setw() and family

//  CODE

namespace xeona
{
  // ---------------------------------------------------------
  //  FREE FUNCTION   : xeona::presentValue
  // ---------------------------------------------------------
```

```
//          FV
//  PV = --------
//              t
//       (1 + i)
//
// The timebase depends on specification of i.

double
presentValue                // PV, year-zero present value
(double futureValue,        // FV, projected future value
 int    timePeriod,         // t, annual with zero-based count
 double interestRate)       // i, interest rate per annum and suitably risk-adjusted
{
  return futureValue / std::pow((1 + interestRate), timePeriod);
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::annuityValue
// ----------------------------------------------------------

//           PV * i
//  A = ---------------
//                1
//      1 -  --------
//                   n
//            (1 + i)
//
// The timebase depends on specification of i.

double
annuityValue                // A, annuity value
(double   presentValue,     // PV, year-zero present value
 unsigned paymentsCount,    // n, number of annual payments
 double   interestRate)     // i, interest rate per annum and suitably risk-adjusted
{
  static logga::spLogger logger = logga::ptrLogStream();

  // defensive programming
  if ( paymentsCount == 0 )
    {
      logger->repx(logga::warn, "invalid payments count, returning 0", paymentsCount);
      const double nan = std::numeric_limits<double>::quiet_NaN();
      return nan;
    }

  // active code
  if ( interestRate == 0 )  // special case of zero interest, otherwise a NaN results
    {
      return presentValue
        / paymentsCount;
    }
  else
    {
      // CAUTION: 'paymentsCount' is 'unsigned', which is unacceptable to 'std::pow'
      return presentValue * interestRate
        / (1 - ( 1 / std::pow((1 + interestRate), static_cast<int>(paymentsCount))));
    }
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::capitalRecovery
// ----------------------------------------------------------
//  Description  : calculates the capital recovery
//  Role         : used for accrued costs reporting
//  Techniques   : uses 'presentValue' and 'annuityValue' functions
//  Status       : complete
//
//  Design notes
//
//      The behavior of this call is identical to the spreadsheet
//      function PMT(rate, periods, investment value, future
//      value).
//
//      The (optional argument) current age will set the cost
//      stream to zero if the current age is equal to or greater
//      than the economic life.
//
//  Financial equations sourced from Wikipedia
//
//      wikipedia page title: Time value of money
```

```
  //      retrieved from: http://en.wikipedia.org/wiki/Time_value_of_money
  //      page last modified: 26-Mar-2008 at 19:36
  //      page still present: 31-Jul-2010
  //
  //  Comparative testing
  //
  //      OpenOffice.org Calc 2.0
  //      OpenOffice.org Calc 3.2
  //
  //      PMT : regular payments : returns the periodic payment of an annuity,
  //      based on regular payments and a fixed periodic interest rate
  //
  //      PMT(0.1;         # Rate          : the rate of interest per period
  //          3;           # NPER          : payment period
  //          100;         # PV            : present value
  //          -10)         # FV (optional) : future value
  //
  //      -37.19033233
  //
  //      This uses the default optional Type of 0 meaning
  //      payment occurs at the end of each period.
  //
  // ------------------------------------------------------

  double
  capitalRecovery                  // relative to the length of the time horizon interval
  (double   discountRate,          // risk-adjusted interest per annum (decimal not %age)
   unsigned economicLife,          // economic life in years (not seconds)
   double   capitalInputInitial,   // capital input at beginning of economic life
   double   capitalInputTerminal,  // capital input at end of economic life (positive if a
   unsigned currentAge)            //                                          liability)
  {
    static logga::spLogger logger = logga::ptrLogStream();
    logger->repx(logga::adhc, "entering member function", "");

    const double nan       = std::numeric_limits<double>::quiet_NaN();
    double capitalRecovery = nan;            // initially set to nonsensical value

    if ( economicLife == 0 )                 // xeona convention for disabling the
      {                                      //    calculation, this test also prevents
        capitalRecovery = 0.0;               //   'annuityValue' from choking
      }
    else if ( currentAge >= economicLife )   // asset has been paid off
      {
        capitalRecovery = 0.0;
      }
    else                                     // normal calculation
      {
        // CAUTION: period counts are zero-based and hence the
        // economic life value was originally decremented when used
        // as an index.  However this lead to behavior different to
        // the spreadsheet function PMT and was removed.  In other
        // words, the terminal payment has no associated trailing
        // year or annuity payment.

        double PV
          = presentValue( capitalInputInitial,  0,            discountRate)
          + presentValue( capitalInputTerminal, economicLife, discountRate);

        double A = annuityValue(PV, economicLife, discountRate);

        capitalRecovery = A;
      }

#ifndef _XUTEST
    const int length               = Entity::getHorizonInterval();
#else                                        // to reduce dependencies
    const int length               = 1800;  // half hour
#endif // XE_XUTEST

    double capitalRecoveryPerSecond = capitalRecovery / (60 * 60 * 8760);
    double capitalRecoveryPerInt    = capitalRecoveryPerSecond * length;

    // report at the lowest priority
    // a 'width' of 14 aligns up to one thousand million at precision 3

#ifndef _XUTEST
    const int width     = 14;
    const int precision =  3;
#else
    const int width     = 13;
```

```
    const int precision =  8;               // for testing purposes
#endif // XE_XUTEST

      std::ostringstream put;
    put << std::fixed << std::setprecision(precision);
    put << "  capital recovery calculations"                          << "\n"
        << "    project discount rate [-/y]                  : "
        << std::setw(width) << std::right << discountRate            << "\n"
        << "    current age   [y]                            : "
        << std::setw(width) << std::right << currentAge              << "\n"
        << "    economic life [y]                            : "
        << std::setw(width) << std::right << economicLife            << "\n"
        << "    capital inflow initial  [$] (investment)     : "
        << std::setw(width) << std::right << capitalInputInitial     << "\n"
        << "    capital inflow terminal [$] (decommissioning) : "
        << std::setw(width) << std::right << capitalInputTerminal    << "\n"
        << "    interval length [s]                          : "
        << std::setw(width) << std::right << length                  << "\n"
        << "    capital recovery per annum [$/y]             : "
        << std::setw(width) << std::right << capitalRecovery         << "\n"
        << "    capital recovery per interval [$/interval]   : "
        << std::setw(width) << std::right << capitalRecoveryPerInt   << "\n";

    logger->repx(logga::adhc, "capital recovery values follow", "");
    logger->putx(logga::adhc, put);
    logger->addSmartBlank(logga::adhc);

    return capitalRecoveryPerInt;
  }

} // namespace xeona

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : greenhouse.cc
//  file-create-date : Mon 12-Oct-2009 09:10 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : global warming potential support / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/ghouse.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "ghouse.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting

//  CODE

// ----------------------------------------------------------
//  CLASS           : Gwp100Bundle
// ----------------------------------------------------------
//
//  Abbreviations
//
//      co2      carbon dioxide
//      ch4      methane
//      n2o      nitrous oxide
//
//  100 year GWP (global warming potential)
//
//      Data from wikipedia page at
//      'http://en.wikipedia.org/wiki/Global_warming_potential'
//      quoting 2007 IPCC AR4 p212 (more specifically the 2007
//      IPCC Fourth Assessment Report by Working Group 1 and
//      Chapter 2 of that report "Changes in Atmospheric
//      Constituents and in Radiative Forcing" which contains
//      relevant GWP information).
//
// ----------------------------------------------------------

// STATIC DEFINITIONS

const double Gwp100Bundle::s_gwpCo2    =   1.0;   // necessarily unity
const double Gwp100Bundle::s_gwpCh4    =  25.0;
const double Gwp100Bundle::s_gwpN2o    = 298.0;
```

```
logga::spLogger Gwp100Bundle::s_logger = logga::ptrLogStream();

// CREATORS

Gwp100Bundle::Gwp100Bundle() :
  co2(0.0),
  ch4(0.0),
  n2o(0.0)
{ }

Gwp100Bundle::Gwp100Bundle                    // up-front constructor
(const double a_co2,
 const double a_ch4,
 const double a_n2o) :
  co2(a_co2),
  ch4(a_ch4),
  n2o(a_n2o)
{ }

Gwp100Bundle::Gwp100Bundle                    // tuple constructor
(const tupleD3 tup) :
  co2(tup.get<0>()),
  ch4(tup.get<1>()),
  n2o(tup.get<2>())
{ }

Gwp100Bundle::~Gwp100Bundle() { }              // destructor

// UNARY OPERATORS

Gwp100Bundle&
Gwp100Bundle::operator*=
(const double& other)
{
  co2 *= other;
  ch4 *= other;
  n2o *= other;
  return *this;
}

// ACCESSORS

tupleD3
Gwp100Bundle::tuple() const                   // export 'Gwp100Bundle' to six tuple
{
  return boost::make_tuple(co2, ch4, n2o);
}

double
Gwp100Bundle::totalMass() const
{
  const double mass
    = co2
    + ch4
    + n2o;
  return mass;
}

double
Gwp100Bundle::co2Equivalent() const
{
  const double co2e
    = s_gwpCo2   * co2
    + s_gwpCh4   * ch4
    + s_gwpN2o   * n2o;
  return co2e;
}

double
Gwp100Bundle::gwpEffective() const
{
  return co2Equivalent() / totalMass();
}

// STATIC ACCESSORS

double Gwp100Bundle::getGwpCo2() { return s_gwpCo2; }
double Gwp100Bundle::getGwpCh4() { return s_gwpCh4; }
double Gwp100Bundle::getGwpN2o() { return s_gwpN2o; }
```

```
// MANIPULATORS

void
Gwp100Bundle::reset
(const double value)                          // note the default (of zero)
{
  // reporting
  if ( xeona::releaseStatus == true )
    {
      s_logger->repx(logga::warn, "call intended for development only", "");
    }
  if ( value == 0.0 )
    {
      s_logger->repx(logga::adhc, "reset value", value);
    }
  else
    {
      s_logger->repx(logga::warn, "non-zero reset value", value);
    }

  // active code
  co2   = value;
  ch4   = value;
  n2o   = value;
}

void Gwp100Bundle::setCo2  (const double a_co2) { co2 = a_co2; }
void Gwp100Bundle::setCh4  (const double a_ch4) { ch4 = a_ch4; }
void Gwp100Bundle::setN2o  (const double a_n2o) { n2o = a_n2o; }

// DISPLAY CALLS

std::string
Gwp100Bundle::summarizeMe
(std::string msg) const
{
#if 1 // 0 = ignore code, 1 = enact code
  if ( msg.empty() ) msg = "(no message)";
#endif // 0

  std::ostringstream oss;
  oss << "  GWP gas bundle:" << "\n";
  oss << boost::format("    bundle [kg] : CO2 = %.3g") % co2
      << boost::format("    CH4 = %.3g")               % ch4
      << boost::format("    N2O = %.3g")               % n2o
      << "\n";
  if ( ! msg.empty() )
    {
      oss <<           "    message (options)   : " << msg << "\n";
    }
  oss << boost::format("    total mass [kg]     : %10.6g\n") % totalMass()
      << boost::format("    CO2 equivalent [kg] : %10.6g\n") % co2Equivalent()
      << boost::format("    effective GWP [-]   : %10.6g\n") % gwpEffective();
  return oss.str();
}

std::string
Gwp100Bundle::displayGwps()                   // static
{
  const std::string msg = "hardcoded IPCC 100-year global warming potentials [-]";
  std::ostringstream oss;
  oss << boost::format("  %s:\n")                                              % msg
      << boost::format("    CO2   (carbon dioxide) (unity)  : %10.6g\n") % s_gwpCo2
      << boost::format("    CH4   (methane)                 : %10.6g\n") % s_gwpCh4
      << boost::format("    N2O   (nitrous oxide)           : %10.6g\n") % s_gwpN2o;
  return oss.str();
}

// ------------------------------------------------------
//  FREE FUNCTION   : operator* (const double&,       const Gwp100Bundle&);
//  FREE FUNCTION   : operator* (const Gwp100Bundle&, const double&);
// ------------------------------------------------------

const Gwp100Bundle
operator*
(const double&       lhs,
 const Gwp100Bundle& rhs)
{
  Gwp100Bundle tmp;
  tmp.co2   = lhs * rhs.co2;
  tmp.ch4   = lhs * rhs.ch4;
```

```
  tmp.n2o   = lhs * rhs.n2o;
  return tmp;
}

const Gwp100Bundle
operator*
(const Gwp100Bundle& lhs,
 const double&       rhs)
{
  return operator*(rhs, lhs);
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::totalMass
// ----------------------------------------------------------

namespace xeona
{
  double
  totalMass(const double co2,
            const double ch4,
            const double n2o)
  {
    Gwp100Bundle gwp(co2, ch4, n2o);
    const double mass = gwp.totalMass();
    return mass;
  }
} // namespace 'xeona'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::co2Equivalent
// ----------------------------------------------------------

namespace xeona
{
  double
  co2Equivalent(const double co2,
                const double ch4,
                const double n2o)
  {
    Gwp100Bundle gwp(co2, ch4, n2o);
    const double co2e = gwp.co2Equivalent();
    return co2e;
  }
} // namespace 'xeona'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::gwpEffective
// ----------------------------------------------------------

namespace xeona
{
  double
  gwpEffective(const double co2,
               const double ch4,
               const double n2o)
  {
    Gwp100Bundle gwp(co2, ch4, n2o);
    const double gwpe = gwp.gwpEffective();
    return gwpe;
  }
} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : inbuilt.cc
//  file-create-date : Mon 14-Jan-2008 15:29 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : compiled-in test file generation / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/inbuilt.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "inbuilt.h"          // companion header for this file (place first)

#include "../c/xemgen.h"      // class to generate well-formatted XEM models
#include "../a/xemopt.h"      // skeleton xem model generator
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <fstream>            // file-based io
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ------------------------------------------------------------
//  FREE FUNCTION   : ::quot
// ------------------------------------------------------------

namespace
{
  std::string
  quot
  (const std::string& str)
  {
    return xeona::modelStringDelim + str + xeona::modelStringDelim;
  }
} // unnamed namespace

namespace xeona
{
  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::createModelName
  // ------------------------------------------------------------

  std::string                                 // often leaf but directory part will remain
  createModelName
  (const std::string& stub)                   // empty string uses inbuilt default
  {
    std::string buf(stub);
    if ( boost::ends_with(buf, xeona::modelExt) )
      {
        boost::erase_tail(buf, xeona::modelExt.length());
```

```
      }
    if ( buf.empty() )
      {
        buf = xeona::modelInbuiltDefault;    // as set in "common.cc"
      }
    buf += xeona::modelExt;
    return buf;
  }

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::dumpToFile
  // ------------------------------------------------------------

  bool                                       // 'false' means file open failed
  dumpToFile                                 // calls 'loadOstream' internally
  (const std::string& filename,              // will overwrite file, maybe with zero bytes
   const unsigned     steps)                 // horizon steps
  {
    static logga::spLogger logger = logga::ptrLogStream();

    std::ofstream ofile(filename.c_str());   // note defaults: text, trunc (overwrite)
    if ( ! ofile )
      {
        logger->repx(logga::warn, "failed to open file for writing", filename);
        return false;
      }
    logger->repx(logga::dbug, "opened file for writing", filename);

    xeona::loadOstream(ofile, steps);        // defined below
    return true;

  } // 'ofile' closes on block exit

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::loadOstream
  // ------------------------------------------------------------
  //
  // prior to r2765 this function used to contain:
  //
  //    * a preprocessor macro 'XE_SIMPLIFIED' to control model depth
  //    * a set of outstream to build the model
  //
  // ------------------------------------------------------------

  void
  loadOstream                                // helper, func, contains model in text form
  (std::ostream&  os,
   const unsigned steps)                     // horizon steps
  {
    static logga::spLogger logger = logga::ptrLogStream();

    // create and fill a 'Xem' object, see unit 'a/xemopt'
    Xem xem(os,                              // print to stream
            xeona::svnRev,                   // svn revison
            xeona::modelAngleIndent);        // alignment tab

    xem.head();
    xem.mand(6);                             // number of steps
    xem.rule("model");
    xem.more();
    xem.tail();
    xem.flush();

    logger->repx(logga::dbug, "text streamed to file", "");
  }

} // namespace xeona

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : label.cc
//  file-create-date : Fri 24-Oct-2008 12:18 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : helper class for solver labels  / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/label.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "label.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ----------------------------------------------------------
//  CLASS           : Label
// ----------------------------------------------------------
//  Description  : helper class to collect and format OSP and solver labels
//  Role         : simplify label processing in OSP upload calls
//  Techniques   : overloaded function call operator, output streaming
//  Status       : complete
// ----------------------------------------------------------

// CREATORS

Label::Label
(const std::string& str) :
  d_separator(xeona::ospTagSep),             // separator string set here
  d_ctorArg(str),
  d_labelettes()                             // empty vector
{
  if ( ! str.empty() ) d_labelettes.push_back(str);
}

Label::~Label()
{
}

// PUBLIC CALLS

void
Label::trim
(int num)
```

```
{
  while ( num-- )
    {
      if ( empty() ) break;                 // member function 'empty'
      d_labelettes.pop_back();              // 'pop_back' does not return the element
    }
}

void
Label::resetToCtor()
{
  d_labelettes.clear();
  if ( ! d_ctorArg.empty() ) d_labelettes.push_back(d_ctorArg);
}

bool                                        // 'true' if added
Label::padd                                 // zero-pad the integer
(const unsigned in,
 const unsigned pad)                        // pad level, note default
{
  std::ostringstream oss;
  oss << std::setfill('0') << std::setw(pad) << in;
  const std::string sin = oss.str();
  return add(sin);
}

std::string
Label::str() const                          // principal stringify function
{
  std::string buffer = "";
  BOOST_FOREACH( std::string s, d_labelettes )
    {
      buffer += s;
      buffer += d_separator;
    }
  int buflen = buffer.length();
  int seplen = d_separator.length();
  if ( buflen > 0 ) buffer.erase(buflen - seplen);     // the remove final separator
  return buffer;
}

std::string
Label::padstr                               // zero-pad the integer
(const unsigned in,
 const unsigned pad)                        // pad level, note default
{
  const bool added   = padd(in, pad);       // element inserted
  std::string buffer = str();               // overloaded member function 'str()'
  if ( added ) trim(1);                     // element removed
  return buffer;
}

bool
Label::empty() const
{
  return d_labelettes.empty();
}

// EXPLICIT TEMPLATE SPECIALIZATIONS (TYPE DEDUCED)

// These are type-deduced template specializations, as opposed to
// explicit template instantiations (noting that the latter are
// more often used in this codebase).

template <>
void
Label::operator<<                           // streams as per any default ostream
(const int& in)
{
  padd(in);                                 // padded add
}

template <>
void
Label::operator<<                           // streams as per any default ostream
(const unsigned& in)
{
  padd(in);                                 // padded add
}

// STREAMING SUPPORT
```

```
// ------------------------------------------------------
//  FREE FUNCTION   : operator<< (std::ostream&, Label&)
// ------------------------------------------------------

std::ostream&
operator<<
(std::ostream& os,
 const Label&  label)
{
  os << label.str();
  return os;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : linklog.cc
//  file-create-date : Thu 30-Jul-2009 21:25 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : utility class to record entity linking results / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/linklog.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "linklog.h"          // companion header for this file (place first)

#include "../b/entity.h"      // entity base class plus lazy linking

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/format.hpp>   // printf style formatting
#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ------------------------------------------------------------
//  MEMBER FUNCTION : LinkLogger
// ------------------------------------------------------------

LinkLogger::LinkLogger() :
  d_linkNulls(0),
  d_linkLog()                                 // empty vector
{
  reset();
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : ~LinkLogger
// ------------------------------------------------------------

LinkLogger::~LinkLogger()
{
}

// ------------------------------------------------------------
//  MEMBER FUNCTION : recover
// ------------------------------------------------------------

std::string
```

```
LinkLogger::recover
(const int tab) const
{
  if ( getLogCount() == 0 )                  // bail out if no entries
    {
      return "";
    }

  // CAUTION: tuple member function 'get<0>' works for Boost
  // library 1.35.0 and better, but not for g++ 4.1.2

  std::string buffer;                        // return buffer
  const std::string padding(tab, ' ');       // padding
  BOOST_FOREACH( entry_type entry, d_linkLog )
    {
      buffer += boost::str(boost::format("%s%-30s  %-20s  %-20s  %-15s  %s\n")
                           % padding
                           % entry.get<0>()
                           % entry.get<1>()
                           % entry.get<2>()
                           % entry.get<3>()
                           % entry.get<4>());
    }
  return buffer;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getNullCount
// ----------------------------------------------------------

int
LinkLogger::getNullCount() const
{
  return d_linkNulls;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLogCount
// ----------------------------------------------------------

int
LinkLogger::getLogCount() const
{
  return d_linkLog.size() - 1;               // minus one for header
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : reset
// ----------------------------------------------------------

void
LinkLogger::reset()
{
  d_linkLog.clear();

  // add header using the same formatting calls as entries
  d_linkLog.push_back(boost::make_tuple("identifier",
                                        "full",
                                        "link",
                                        "resource",
                                        "remap"));
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : insert
// ----------------------------------------------------------

bool
LinkLogger::insert
(const xeona::assign_ptr<Entity>& link,
 const bool                       okay)
{
  // count failures
  if ( okay == false ) d_linkNulls++;

  // harvest information
  const std::string identifier    = link->getIdentifier();

#ifndef _XUTEST
  const shared_ptr<Entity> entity = Entity::retSharedPtr(identifier);
  const std::string asBuiltType   = entity->getTypeStr();
```

```
  std::ostringstream oss;
  oss << entity;                              // stream to recover address
  const std::string resource = oss.str();
#else
  const std::string asBuiltType   = "(not under unit test)";
  const std::string resource      = "(not under unit test)";
#endif // _XUTEST

  const std::string linkType      = xeona::demangle(typeid(*link).name());
  const std::string output        = okay ? "okay" : "FAIL";

  // insert data
  d_linkLog.push_back(boost::make_tuple(identifier,
                                        asBuiltType,
                                        linkType,
                                        resource,
                                        output));
  return okay;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : recset.cc
//  file-create-date : Tue 09-Oct-2007 17:14 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : records and fields and also record-sets / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/recset.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "recset.h"           // companion header for this file (place first)

#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../b/entity.h"      // entity base class
#include "../a/exapp.h"       // application exception classes
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting
#include <boost/lexical_cast.hpp>       // lexical_cast<> string to number conversions

//  CODE

// ----------------------------------------------------------
//  notes           : Data in general
// ----------------------------------------------------------
//
//  Record sets
//
//      A 'record set' is a collection of records.
//
//      Currently, only one record set is supported -- that
//      created at start-up.  However, an interactive simulation
//      would need to create and process additional record sets
//      during run-time.
//
//  Values
//
//      Values are the primitives that underpin much of this, or
//      for that matter, any simulation model.  In this
//      documentation, 'values' break down into 'singles' and
//      'timeseries'.  And consequently timeseries contain
//      'elements' and not values.
//
//      No special naming convention in the fieldname
```

```
//      distinguishes singles from timeseries during data input.
//      And a horizon of unity is legal.  However the
//      'modelTsRepeater' string literal (defined as ".." at the
//      time of writing) enables repetition:
//
//          my_single      > 1
//          my_timeseries  > 2 ..
//
//      A value is typically read in as a single trimmed string.
//      This string is then split on whitespace (meaning ONE OR
//      MORE space chars) and stored in a vector:
//
//          input --+-- raw string   : std::string
//                  |
//                  +-- split string : std::vector<std::string>
//
//      If the split string is not of the required length, it may
//      be truncated or repeated (or repeated and truncated) to
//      suit (similar to vector "recycling" in the R language).
//
//      For reasons of simplicity, vector's are often used to
//      hold values.  Therefore, a single is a special case
//      timeseries with just one element (not an unreasonable
//      abstraction).
//
//          value --+-- single     : single element std::vector
//                  |
//                  +-- timeseries : multi-element std::vector
//
//      The entity itself is responsible for determining the
//      underlying type and can elect to use anything supported
//      by boost::lexical_cast<> and able to be stringified by
//      boost::format or the stream inserter operator<<
//      (including, for instance, a long double).  In addition,
//      the relevant typedefs must also be modified.
//
//      The following value data types are supported in this
//      simulation model:
//
//          value --+-- number  : int, double
//                  |
//                  +-- boolean : bool, boost::logic::tribool
//                  |
//                  +-- text    : std::string
//
//      Values may also play one of two roles, thus:
//
//          value --+-- input  : externally supplied
//                  |
//                  +-- output : internally generated
//
//      Singles objects also need to be "wrapped" so that they
//      can initialize entity data members held as references.
//      See the comments pertaining to class 'wrap' for details.
//
// ------------------------------------------------------------

// ------------------------------------------------------------
//  CLASS (FUNCTOR) : ::UnlexElement <>
// ------------------------------------------------------------
//  Description  : interface to 'UnlexSingle' with 'wrap' transformation added
//  Supports     : 'TimeseriesUnlex_Helper'
//  Uses         : 'UnlexSingle'
//  Status       : complete
// ------------------------------------------------------------

namespace
{
  class UnlexSingle;                        // forward declaration

  template <typename T>
  class UnlexElement
  {
  public:

    std::string
    operator()
    (T element)                             // T must be supported by single_type
    {
      ::UnlexSingle um;                     // functor
      return um(wrap<T>(element));          // uses 'single_type', hence the "conversion"
```

```
    }

  };

} // unnamed namespace

// ----------------------------------------------------------
//  FREE FUNCTION   : ::TimeseriesUnlex_Helper <>
// ----------------------------------------------------------
//  Description  : provides generic 'timeseries' traversal routines
//  Supports     : 'TimeseriesUnlex'
//  Uses         : 'UnlexElement'
//  Status       : complete
// ----------------------------------------------------------

namespace
{
  template <typename T>
  std::vector<std::string>
  TimeseriesUnlex_Helper
  (const shared_ptr<std::vector<T> > input)
  {
    unsigned horizon = Entity::getHorizonSteps(); // grab horizon steps
    std::vector<std::string> output(horizon);

    std::transform                          // copy and modify, requires <algorithm>
      (input->begin(),                      // source range start
       input->end(),                        // source range end
       output.begin(),                      // destination range start
       ::UnlexElement<T>());                // unitary operation, '::' optional

    return output;
  }

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS (FUNCTOR) : ::UnlexTimeseries_VariantVisitor <>
// ----------------------------------------------------------
//  Description  : boost::variant functor for unlexing 'timeseries'
//  Supports     : 'unlexTimeseries'
//  Uses         : 'TimeseriesUnlex_Helper'
//  Techniques   : Boost.Variant visitor
//  Status       : complete
// ----------------------------------------------------------

namespace
{

  class UnlexTimeseries_VariantVisitor :
    public boost::static_visitor<std::vector<std::string> >
  {

    typedef std::vector<std::string> return_type;  // CAUTION: same as line above

  public:

    //int, double, bool, boost::logic::tribool, std::string

    return_type operator() (const shared_ptr<std::vector<int> > input) const
    {
      return ::TimeseriesUnlex_Helper<int>(input);
    }

    return_type operator() (const shared_ptr<std::vector<double> > input) const
    {
      return ::TimeseriesUnlex_Helper<double>(input);
    }

    return_type operator() (const shared_ptr<std::vector<bool> > input) const
    {
      return ::TimeseriesUnlex_Helper<bool>(input);
    }

    return_type operator() (const shared_ptr<std::vector<tribool> > input) const
    {
      return ::TimeseriesUnlex_Helper<tribool>(input);
    }

    return_type operator() (const shared_ptr<std::vector<std::string> > input) const
    {
```

```
      return ::TimeseriesUnlex_Helper<std::string>(input);
    }

  };

} // unnamed namespace

// ------------------------------------------------------
//  CLASS (FUNCTOR) : ::UnlexElement_VariantVisitor
// ------------------------------------------------------
//  Description  : boost::variant functor returning formatted string
//  Role         : processing of boost::variant fields for output
//  Note         : formats output for both 'singles' and 'timeseries'
//  Supports     : 'UnlexSingle'
//  Uses         : call chain end
//  Techniques   : Boost.Variant visitor and Boost.Format library
//  Status       : complete
//
//  Design notes
//
//      The use of call-by-reference (rather than call-by-value)
//      in operator() prevents implicit conversions, for instance
//      from char to int (see Karlsson 2006).
//
//  Boost.Format
//
//      Boost.Format syntax (with the positional format
//      specification omitted)
//
//          %[flags][width][.precision]type-char
//
//          flags     : - + _ + # 0 [spc]
//          type-char : px, o, e, f, g, diu, s, c, % (also some uppercase)
//
//      Output formatting options used here
//
//          double  : %e    7 sig-figs, always scientific
//                    %g    6 sig-figs, scientific or raw (default)
//                    %.2e  3 sig-figs, scientific
//          int     : %i    straight output (equivalent to %d)
//
//          bool    : std::noboolalpha  0 or 1
//          tribool : std::noboolalpha  0, 1, 2
//
//          string  : %s  (can set precision first chars, also flags and width)
//
//  CAUTION: setting tribools
//
//     Set tribools with key-words (false, true, indeterminate)
//     and not integer literals (0, 1, 2).
//
//  See also
//
//      Karlsson (2006 pp191-207) describes the Boost.Variant
//      library.
//
//      The Boost.Format library is not covered in text books as
//      far as I know.  See the Boost documentation instead:
//
//        file:///path-to-boost/boost_1_34_1/libs/format/index.html
//
// ------------------------------------------------------

// NOTE: this functor works equally with 'int' and 'wrap<int>'
// objects without modification

namespace
{
  class UnlexElement_VariantVisitor :
    public boost::static_visitor<std::string>
  {
  public:

    //int, double, bool, boost::logic::tribool, std::string

    std::string operator() (const int& i) const
    {
      return boost::str(boost::format("%i") % i);
    }

    std::string operator() (const double& d) const
```

```
    {
      // should stabilize on "%e"
      // return boost::str(boost::format("%e") % d);
      // return boost::str(boost::format("%g") % d);
      // return boost::str(boost::format("%.2e") % d);
      return boost::str(boost::format("%+.2e") % d);   // always sci, leading +, 3 sigfig
    }

    std::string operator() (const bool& b) const
    {
      std::ostringstream os;
      os << std::noboolalpha << b;
      return os.str();
    }

    std::string operator() (const tribool& t) const
    {
      std::ostringstream os;
      os << std::noboolalpha << t;
      return os.str();
    }
    std::string operator() (const std::string& s) const
    {
      const std::string del = xeona::modelStringDelim;
      return boost::str(boost::format("%s%s%s") % del % s % del);
    }

  };

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS (FUNCTOR) : ::UnlexSingle
// ----------------------------------------------------------
//  Description  : format a variant type and return a std::string
//  Role         : unitary operator for STL container algorithm
//  Supports     : 'unlexSingle', 'UnlexElement'
//  Uses         : 'UnlexElement_VariantVisitor'
//  Techniques   : Boost.Variant library
//  Status       : complete
//
//  Design notes
//
//      Please note that 'UnlexElement_VariantVisitor' could not
//      be applied directly because of the boost::apply_visitor
//      syntax.  Perhaps I got this wrong but I tried pretty hard
//      to rationalize the call chains used here.  Hence the need
//      for this wrapper functor (which is analogous to
//      'LexElement' in any case).  [If interested in developing
//      this code, check out Karlsson (2006 p205)].
//
// ----------------------------------------------------------

namespace
{
  class UnlexSingle
  {
    // CAUTION: 'single_type' must be visible here (probably
    // because it has been declared as a global 'typedef' in the
    // header file).

  public:

    std::string
    operator()
    (single_type element)                          // CAUTION: 'const' caused problems
    {
      ::UnlexElement_VariantVisitor sf;            // formatting functor
      return boost::apply_visitor(sf, element);    // automatically selects correct type
    }

  };

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS (FUNCTOR) : ::LexElement <>
// ----------------------------------------------------------
//  Description  : cast from std::string to nominated type T
//  Role         : unitary operator for STL container algorithm
//  Supports     : 'lexValue'
```

```
//  Uses         : call chain end
//  Techniques   : Boost.Conversion library
//  Status       : complete
// ---------------------------------------------------------

namespace
{
  template <typename T>
  class LexElement
  {
  public:

    LexElement() : d_loggerCalls(0) { }      // explicitly initialize 'd_loggerCalls'

    T                                        // functor return type is T
    operator()                               // CAUTION: template keyword not required
    (const std::string& element)
    {
    try
      {
        return boost::lexical_cast<T>(element);   // Boost.Conversion library
      }
    catch( const boost::bad_lexical_cast& eblc )
      {
        if ( d_loggerCalls < xeona::sameLogLimit )
          {
            std::ostringstream put;
            put << "    lexical cast failure: " << eblc.what() << "\n";
            s_logger->putx(logga::dbug, put);
            std::string quoteElem = '"' + element + '"';
            s_logger->repx(logga::warn, "numeric cast type incompatibility", quoteElem);
          }
        else if ( d_loggerCalls == xeona::sameLogLimit )
          {
            s_logger->repx(logga::info, "same log limit exceeded", xeona::sameLogLimit);
          }
        d_loggerCalls++;
        return T();                          // zero-argument (default) construction
      }
    }

  private:

    unsigned                 d_loggerCalls; // to prevent a deluge of log messages
    static logga::spLogger   s_logger;      // shared_ptr to single logger object

  };

  //  STATIC DEFINITIONS

  template <typename T>
  logga::spLogger
  LexElement<T>::s_logger = logga::ptrLogStream();   // bind logger on definition

} // unnamed namespace

// ---------------------------------------------------------
//  CLASS           : Field
// ---------------------------------------------------------
//  Description  : holds field information
//  Role         : used directly or by friendly Record objects
//  Status       : complete
//
//  Design notes
//
//      A Field object, after being loaded from a model file,
//      usually holds either one string element or an entire time
//      horizon of separated string elements in 'd_splitStr'.
//
//      These strings, however, are not lexically converted until
//      the data has been requested by the associated entity --
//      at which time the underlying type is known.  This
//      conversion process is managed by the host Record object
//      but the modification code is provided by this class.
//
//      The converted data is bound by reference or shared_ptr to
//      the relevant entity.  This means that the record set and
//      the associated entity hold the same data.
//
//      The original string input is retained in 'd_rawStr'.  In
//      actuality (at the time of writing), 'd_rawStr' plays no
```

```
//      role after initialization and could be omitted in the
//      interests of memory conservation.
//
//      This class also relies on a number of helper functors,
//      also defined in this file.
//
//  Note on empty objects
//
//      An empty shared_ptr object returns member function
//      'use_count()' = 0 (whereas a shared_ptr holding a NULL of
//      the correct type yields 'use_count()' = 1 and 'get()' =
//      0).  The statement 'if ( sp )' can be used to determine
//      if the pointer holds anything useful.
//
//      In regards boost::variant, the member function 'empty()'
//      ALWAYS returns false (Karlsson 2006 p195).  This is
//      because the zero-argument (default) boost::variant
//      constructor zero-argument (default) constructs the first
//      type in the list.
//
// ----------------------------------------------------------

//  STATIC DEFINITIONS

logga::spLogger
Field::s_logger = logga::ptrLogStream();     // bind logger on definition

// CREATORS

Field::Field() :
  d_name(),
  d_kind(xeona::e_notField),
  d_enabled(indeterminate),
  d_units(),
  d_remark(),
  d_rawStr(),
  d_splitStr(),
  d_single(),
  d_timeseries()
{ }

// MANIPULATORS -- for loading data

void Field::addName(std::string fieldname)          { d_name       = fieldname; }
void Field::addKind(xeona::FieldKind kind)          { d_kind       = kind;      }
void Field::addEnabled(tribool enabled)             { d_enabled    = enabled;   }
void Field::addUnits(std::string units)             { d_units      = units;     }
void Field::addRemark(const std::string& remark)    { d_remark     = remark;    }
void Field::addRawStr(const std::string& rawStr)    { d_rawStr     = rawStr;    }
void Field::addSplitStr(const std::vector<std::string>& splitStr)
                                                    { d_splitStr   = splitStr;  }
// ACCESSORS

std::string                     Field::getName()     const { return d_name;    }
xeona::FieldKind                Field::getKind()     const { return d_kind;    }
tribool                         Field::getEnabled()  const { return d_enabled; }
std::string                     Field::getUnits()    const { return d_units;   }
std::string                     Field::getRemark()   const { return d_remark;  }

std::string                     Field::getRawStr()   const { return d_rawStr;           }
const std::vector<std::string>& Field::getSplitStr() const { return d_splitStr;         }
int                             Field::getCount()    const { return d_splitStr.size();  }
bool                            Field::isEmpty()     const { return d_splitStr.empty(); }

  // ACCESSORS -- after binding has occurred

const std::string
Field::getSingle()
{
  unlexSingle(d_single);
  return d_splitStr.front();
}

const std::vector<std::string>
Field::getTimeseries()
  throw(xeona::empty_field_on_write)         // exception specification
{
  // protect against 'd_timeseries' remaining default constructed
  // -- meaning that no subsequent reassignment has occurred (the
  // most likely explanation is that dataset never contained this
  // field)
```

```
  try
    {
      typedef shared_ptr<std::vector<int> > default_type;
      default_type test = boost::get<default_type>(d_timeseries);
      if ( test == 0 )
        {
          // 'boost::get' succeeded but shared pointer null or
          // empty -- a aforementioned problem thus identified
          s_logger->repx(logga::warn, "non-reassigned timeseries", d_name);
          if ( xeona::nopro == false )       // meaning option '--krazy' not applied
            {
              s_logger->repx(logga::dbug, "will throw xeona::empty_field_on_write", "");
              throw xeona::empty_field_on_write("timeseries", d_name);
            }
        }
    }
  catch( const boost::bad_get& e )
    {
      // do nothing is correct -- to be here means that
      // 'd_timeseries' is no longer default constructed
    }

// active code
  unlexTimeseries(d_timeseries);
  return d_splitStr;
}

const std::string
Field::getTimeseries
(const std::string& sep)                     // 'sep' overloaded form of 'getTimeseries'
{
  std::string buf;
  BOOST_FOREACH( std::string s, getTimeseries() )
    {
      buf += sep;
      buf += s;
    }
  if ( ! buf.empty() )
    buf = buf.substr(sep.length());          // trim leading 'sep' string (often a space)
  return buf;
}

// MANIPULATORS -- for use by friendly Record objects

void
Field::splitRawStr()                         // process raw string into split string
  throw(std::exception,                      // exception specification
        xeona::short_timeseries)
{
  // PREAMBLE

  // obtain horizon and string delimiter from elsewhere
  unsigned horizon = Entity::getHorizonSteps();   // grab horizon steps
  const char* delim = xeona::modelStringDelim.c_str();      // [1]

  // CAUTION: the Boost.Tokenizer boost::char_separator<char>
  // 'separator' would only accept a C-string -- and not a
  // std::string or char (as might be expected)
  //
  // [1] model string delimiter: the string delimiter 'delim' is
  // defined in "common.cc" and was (at the time of writing) set
  // to an escaped double quote (\") but could equally be a
  // single quote (') or similar.

  // define a string vector buffer
  std::vector<std::string> split;

  // CHECK IF PAIRWISE STRING DELIMITERS ARE PRESENT

  boost::find_all(split, d_rawStr, delim);   // temporarily borrow 'split'
  unsigned delims = split.size();            // number of delims found, if any
  split.clear();                             // return 'split' as found

  if ( delims > 0 && (delims % 2) != 0 )     // inbuilt remainder operator
    {
      std::string msg = getName() + " delims unbalanced";
      s_logger->repx(logga::warn, msg, delims);
      return;
    }

  // UNDERTAKE SPLIT USING APPROPRIATE APPROACH
```

```
  if ( delims == 0 )                        // process as non-strings
    {
      // CAUTION: 'boost::token_compress_on' means "adjacent
      // separators are merged together", otherwise (by default)
      // every pair of separators delimits a token, either empty
      // or full as the case may be

      boost::split(split, d_rawStr, boost::is_any_of(" "), boost::token_compress_on);
    }
  else if ( delims > 0 )                    // process as strings
    {
      // CAUTION: "no parsing is actually done upon construction
      // -- parsing is done on demand as the tokens are accessed
      // via the iterator" (Boost.Tokenizer documentation,
      // 'Tokenizer Class' page)
      //
      // The general approach: the code in the 'for' block is
      // deliberately step-by-step so it can be readily modified

      typedef boost::char_separator<char> separator;
      typedef boost::tokenizer<separator> tokenizer;
      typedef tokenizer::iterator         iterator;

      separator sep(delim);
      tokenizer tokens(d_rawStr, sep);
      for ( iterator it = tokens.begin();
            it         != tokens.end();
            ++it )
        {
          std::string tok(*it);             // grab current token
          if ( tok.empty() ) continue;      // the next line would also catch this
          if ( boost::all(tok, boost::is_space()) ) continue;    // [1]
          boost::trim(tok);                 // trim whitespace from both ends
          split.push_back(tok);             // useful data (we hope)
        }

      // [1] the outer predicate 'boost::all' holds if all
      // elements (chars in this case) satisfy the inner
      // predicate 'boost::is_space'
    }

  // SPECIAL TREATMENT FOR BUILT-IN REMARK

  // this code allows for the non-standard "builtin-remark"
  // values of { null "" "string" }

  if ( getName() == "builtin-remark" )
    {
      split.clear();
      split.push_back("");
    }

  // CHECK AND COMMENT IF DATA IS NOT WELL-FORMED

  if ( split.empty() )
    {
      if ( getEnabled() )                   // skip disabled fields
        {
          s_logger->repx(logga::rankNoData, "enabled field value is empty", getName());
        }
      return;
    }
  else if ( split.front() == xeona::modelTsRepeater )
    {
      std::string repeater = split.front();
      s_logger->repx(logga::warn, "field value is repeat indicator", repeater);
      return;
    }

  // GENERATE A FULL TIMESERIES IF REQUESTED

  if ( split.back() == xeona::modelTsRepeater )
    {
      split.pop_back();                             // remove the repeat indicator
      std::vector<std::string> pattern(split);      // copy construct
      split.empty();
      split.resize(horizon);                        // set split to required length
      xeona::vectorRepeat(pattern, split);          // implicit template instantiation
    }
```

```
  // CHECK FOR AN UNDERSIZE TIMESERIES [1]

  unsigned len = split.size();
  if ( len > 1 && len < horizon )
    {
      int delta = horizon - len;
      std::string message = "'" + getName() + "' short by";
      s_logger->repx(logga::warn, message, delta);

      if ( xeona::nopro == false )
        {
          s_logger->repx(logga::dbug, "will throw xeona::short_timeseries", "");
          throw xeona::short_timeseries(getName(), delta);
        }
    }

  // [1] singles versus timeseries: it is NOT POSSIBLE, at this
  // point, to distinguish between a trivial timeseries with only
  // one element and a genuine single -- the discriminating
  // information is held by entities and any discrepancies will
  // become apparent during entity creation (even in the event
  // that the horizon is set to unity)

  // LOAD INFO

  d_splitStr = split;         // early returns necessarily leave 'd_splitStr' untouched
}

template <typename T>
void
Field::lexValue                                 // string to type T conversion
(shared_ptr<std::vector<T> > output)            // an empty shared_ptr cannot be passed in
{
  // Josuttis (1999 pp367-368) discusses std::transform().  Note
  // that "the caller must ensure the destination range is big
  // enough or that insert iterators are used" (p367).

  unsigned horizon = Entity::getHorizonSteps();   // grab horizon steps
  output->clear();                              // empty the container (if not already so)
  output->resize(horizon);                      // grow using default elements (or shrink)

  std::transform                                // copy and modify, requires <algorithm>
    (d_splitStr.begin(),                        // source range start
     d_splitStr.end(),                          // source range end
     output->begin(),                           // destination range start
     ::LexElement<T>());                        // unitary operation, '::' optional
}

void
Field::unlexSingle
(const single_type& input)
{
  d_splitStr.clear();                           // remove all elements
  ::UnlexSingle um;                             // functor wrapper to formatting functor
  d_splitStr.push_back(um(input));              // the one and only element
}

void
Field::unlexTimeseries
(const timeseries_type input)
{
  d_splitStr.clear();                           // remove all elements
  ::UnlexTimeseries_VariantVisitor tf;
  d_splitStr = boost::apply_visitor(tf, input);
}

// ------------------------------------------------------------
//  CLASS           : Record
// ------------------------------------------------------------
//  Description  : holds a record, comprising metainfo and fields
//  Role         : used by RecordSet
//  Status       : complete
//
// 'Entity' constructor usage for 'tieSingle' and 'tieTimeseries':
//
//      Entity(Record& record) :
//        d_record(record),
//        d_coeff(record.tieSingle<double>("coeff")),    // single value
//        d_count(record.tieTimeseries<int>("count")),   // timeseries value
//        d_local(false)
//      { }
```

```
//
//      Record&           d_record;                 // not strictly necessary
//      double&           d_coeff;
//      std::vector<int>  d_count;
//      bool              d_local;
//
// ----------------------------------------------------------

//  STATIC DEFINITIONS

logga::spLogger Record::s_logger
  = logga::ptrLogStream();                  // bind logger on definition

// CREATORS

Record::Record() :
  d_identifier(),
  d_kind(xeona::e_notRecord),
  d_enabled(indeterminate),
  d_fields(),
  d_comments()
{
  // CAUTION: unsatisfactory interactions between two static
  // objects -- one a static data member of type 'Record' defined
  // by the 'Entity' class and the other the static data member
  // 'Record::s_logger copy assigned from the
  // 'logga::shared_ptr<Logger>' instance held by free function
  // 'logga::ptrLogStream' -- can lead to 'Record::s_logger' not
  // being satisfactorily initialized and hence null.  The order
  // of events must be important -- however I was not able to
  // rectify the problem directly and so the following protection
  // is applied:

  if ( s_logger )                           // a shared pointer
    {
      s_logger->repx(logga::xtra, "constructor call", "");
    }
  else                                      // used in the case just cited
    {
      logga::spLogger logger = logga::ptrLogStream();
      logger->repx(logga::dbug, "constructor call, my s_logger", s_logger.get());
    }
}

Record::~Record()
{
  // CAUTION: 'Logger' object calls placed here will cause
  // trouble if the 'Logger' singleton has already been
  // destroyed, hence the following protection is required (a
  // similar problem exists for the constructor):

  if ( s_logger )                           // a shared pointer
    {
      s_logger->repx(logga::xtra, "destructor call", "");
    }
#ifdef _XUTEST
  else
    {
      // CAUTION: this code will produce reporting after the
      // closing log rule on the console (the final row of dots)
      // and that doesn't look that great on an application build

      logga::spLogger logger = logga::ptrLogStream();
      logger->repx(logga::adhc, "destructor call, my s_logger", s_logger.get());
    }
#endif // 0
}

// MANIPULATORS -- for loading data

void Record::addIdentifier(std::string recordId)    { d_identifier = recordId;       }
void Record::addKind(xeona::RecordKind recordKind)  { d_kind       = recordKind;     }
void Record::addEnabled(tribool recordEnabled)      { d_enabled    = recordEnabled;  }
void Record::addComment(const std::string& comment) { d_comments.push_back(comment); }

void Record::addField(shared_ptr<Field> field)
  throw(std::exception,                     // exception specification
        xeona::xem_data_issue)
{
  // check for duplicate enabled class or data field and throw if encountered
  const std::string name = field->getName();
```

```
  if ( field->getEnabled()                    // CAUTION: tribool is 'true'
       &&                                     // .. meaning skip disabled fields
       locateFieldQuietly(name) )             // tests 'true' if already exists
    {
      const std::string value = field->getRawStr();
      if ( name == "class" )                 // special case of: class >
        {
          s_logger->repx(logga::warn, "duplicate class field encountered", name);
          if ( xeona::nopro == false )       // meaning option '--krazy' not applied
            {
              s_logger->repx(logga::dbug, "will throw xeona::xem_data_issue", "");
              throw xeona::xem_data_issue("duplicate class field encountered",
                                          name,
                                          value);
            }
        }
      else
        {
          s_logger->repx(logga::warn, "duplicate data field encountered", name);
          if ( xeona::nopro == false )
            {
              s_logger->repx(logga::dbug, "will throw xeona::xem_data_issue", "");
              throw xeona::xem_data_issue("duplicate data field encountered",
                                          name,
                                          value);
            }
        }
    }

  // perform update
  d_fields.push_back(field);
}

// ACCESSORS

const std::string               Record::getIdentifier() const { return d_identifier; }
xeona::RecordKind               Record::getKind()       const { return d_kind;       }
tribool                         Record::getEnabled()    const { return d_enabled;    }

const std::vector<std::string>&        Record::getComments() const { return d_comments; }
const std::vector<shared_ptr<Field> >& Record::getFields()   const { return d_fields;   }

const std::string
Record::locateClass() const
{
  const shared_ptr<Field> f = locateField("class");
  if ( ! f )
    {
      s_logger->repx(logga::warn, "unable to locate 'class' for record", d_identifier);
      return "";
    }
  std::string str = f->getRawStr();
  if ( str.empty() )
    {
      s_logger->repx(logga::warn, "empty 'class' string for record", d_identifier);
      return "";
    }
  return str;
}

// BINDING FUNCTIONS -- for use by entities

template <typename T>                         // T must be supported by 'single_type'
wrap<T>                                       // see design notes for class 'wrap'
Record::tieSingle
(const std::string& fieldname)
{
  // locate the required field
  shared_ptr<Field> f;                        // field of interest
  f = locateField(fieldname);                 // called function warns if not found
  if ( ! f )
    return wrap<T>();                         // empty wrap object

  // lex 'splitStr' now that T is known
  shared_ptr<std::vector<T> > lexed(new std::vector<T>());
  f->lexValue<T>(lexed);                      // load temp with type T values

  // create and distribute an aliasable single
  T value = lexed->front();                   // grab the first and only element
  wrap<T> single(value);                      // create "wrapped" form
  f->addSingle<T>(single);                    // copy assign by reference to the recordset
```

```
  return single;                                // return an aliasable object to the entity
}

template <typename T>                           // T in {int,double,bool,tribool,std::string}
wrap<T>                                         // see design notes for class 'wrap'
Record::tieConstant
(const T constantValue)
{
  // create and distribute an aliasable constant
  T value = constantValue;                      // grab the given value
  wrap<T> constant(value);                      // create "wrapped" form
  return constant;                              // return an aliasable object to the entity
}

template <typename T>
shared_ptr<std::vector<T> >                     // only the contained type T is specified
Record::tieTimeseries
(const std::string& fieldname)
  throw(xeona::timeseries_not_found)            // exception specification
{
  // locate the required field
  shared_ptr<Field> f;                          // field of interest
  f = locateField(fieldname);                   // called function warns if not found
  if ( ! f )
    {
      s_logger->repx(logga::warn, "timeseries field not found", fieldname);
      if ( xeona::nopro == false )       // meaning option '--krazy' not applied
        {
          // CAUTION: the following exception is NOT caught
          // despite the presence of suitable catch blocks,
          // moreover this is also a problem if
          // 'std::runtime_error' it thrown instead (a complete
          // mystery)

          const std::string templateType = xeona::demangle(typeid(T).name());
          s_logger->repx(logga::dbug, "will throw xeona::timeseries_not_found", "");
          throw xeona::timeseries_not_found(fieldname, templateType);
        }
      else
        {
          s_logger->repx(logga::dbug, "now running unprotected code", "");
        }

#if 1 // 1 = returns zero-length vector, 0 = returns empty shared pointer (original code)
      s_logger->repx(logga::warn, "probable out-of-range throw shortly", "");
      shared_ptr<std::vector<T> > zeroLengthVector(new std::vector<T>());
      return zeroLengthVector;
#else
      s_logger->repx(logga::warn, "probable segfault shortly", "");
      return typename shared_ptr<std::vector<T> >::shared_ptr();  // empty smart pointer
#endif // 0
    }

  // lex 'splitStr' now T is known
  shared_ptr<std::vector<T> > lexed(new std::vector<T>());
  f->lexValue<T>(lexed);                        // load temp with type T values

  // distribute the smart pointer object
  f->addTimeseries<T>(lexed);                   // smart pointer copy to the recordset
  return lexed;                                 // return a shared pointer to the entity
}

const shared_ptr<Field>                         // returns empty smart pointer if not found,
Record::locateField                             //   also logs a warning
(const std::string& fieldname) const
{
  shared_ptr<Field> ret = locateFieldQuietly(fieldname);
  if ( ! ret )                                  // test for failure
    {
      s_logger->repx(logga::warn, "field name not found", fieldname);
    }
  else
    {
      s_logger->repx(logga::adhc, "field name found", fieldname);
    }
  return ret;                                   // empty or full smart pointer
}

const shared_ptr<Field>                         // returns empty smart pointer if not found,
Record::locateFieldQuietly                      //   but no logging occurs
(const std::string& fieldname) const
```

```
{
  BOOST_FOREACH( shared_ptr<Field> f, d_fields )
    if ( f->getName() == fieldname )
      return f;                                 // full smart pointer
  return shared_ptr<Field>::shared_ptr();     // empty smart pointer
}

void
Record::processFields()
{
  BOOST_FOREACH( shared_ptr<Field> f, d_fields )
    {
      f->splitRawStr();
    }
}

// MANIPULATORS -- for unit tests which make entities but do not read from a '.xem' file

// ------------------------------------------------------------
//  MEMBER FUNCTION : Reccord::hackUnitTestRecord
// ------------------------------------------------------------
//  Description  : modify a dummy 'Record' object to be okay for a 'FullEntity' instance
//  Role         : unit tests which do not read from a model (*.xem) file
//  Requires     : the '_XUTEST' macro must be set for a meaningful call
//  Caller       : 'Record' instance within a unit test test block
//  Status       : complete
// ------------------------------------------------------------

void
Record::hackUnitTestRecord                    // CAUTION: hollow function unless '_XUTEST'
(const std::string builtinRemark,
 const int         horizonSteps)
{
#ifndef _XUTEST

  s_logger->repx(logga::warn, "hollow function as not '_XUTEST'", "");
  return;

#else

  // hardcoded constants
  const std::string fieldname       = "builtin-remark";     // given in 'FullEntity'
  const int         horizonInterval = 3600;                 // usual one hour

  // some reporting
  std::ostringstream oss;
  oss << "\"" << builtinRemark << "\"";
  s_logger->repx(logga::dbug, "entering member function with remark", oss.str());
  if ( builtinRemark.empty() )
    s_logger->repx(logga::info, "empty remark supplied", builtinRemark);

  // main work
  shared_ptr<Field> fBir(new Field());       // new 'Field' instance
  fBir->addName(fieldname);
  fBir->addKind(xeona::e_output);
  fBir->addEnabled(true);
  fBir->addRawStr(builtinRemark);            // from the function argument
  addField(fBir);                            // insert field

  // set some necessary values directly
  Entity::setHorizonDirectly(horizonSteps, horizonInterval);     // [1]
  //[1]: entire func in '_XUTEST'

  // CAUTION: the following call should not be used: fBir->splitRawStr();

  // some testing and reporting
  const shared_ptr<Field> fRem = locateField(fieldname);
  if (! fRem )
    {
      s_logger->repx(logga::warn, "field not located, fieldname", fieldname);
    }
  else
    {
      std::ostringstream put;
      put << std::boolalpha
          << "  field satisfactorily located"                              << "\n"
          << "    name      : "   << fRem->getName()                       << "\n"
          << "    kind      : "   << fRem->getKind()                       << "\n"
          << "    enabled   : "   << fRem->getEnabled()                    << "\n"
          << "    empty     : "   << fRem->isEmpty()                       << "\n"
          << "    value     : \"" << fRem->getRawStr() << "\""             << "\n"
```

```
          << "    split cnt : "   << fRem->getCount()                      << "\n"
          << "  horizon details"                                           << "\n"
          << "    steps     : "   << Entity::getHorizonSteps()             << "\n"
          << "    interval  : "   << Entity::getHorizonInterval()          << "\n"
        ; // trailing semicolon
      s_logger->putx(logga::adhc, put);      // reporting threshold set high
    }
  return;

#endif // _XUTEST
}

// ----------------------------------------------------------
//  CLASS (FUNCTOR) : ::IfRecKind
// ----------------------------------------------------------
//  Description  : boolean functor for 'RecordSet'
//  Role         : 'std::remove_copy_if' algorithm predicate in 'copySubset'
//  Status       : complete
//
//  Design notes
//
//      Note carefully that the predicate logic is "remove if
//      true" and NOT "copy if true" -- for some reason there is
//      no 'std::copy_if' algorithm.
//
// ----------------------------------------------------------

namespace
{
  class IfRecKind
  {
  private:                                  // public copy constructor required

    IfRecKind();                                        // zero-argument constructor
    IfRecKind& operator= (const IfRecKind& orig);       // copy assignment operator

  public:

    IfRecKind
    (const xeona::RecordKind recKind) :
      d_recKind(recKind)
    {
      logga::spLogger logger = logga::ptrLogStream();
      logger->repx(logga::dbug, "constructor call (functor)", "");
    }

    bool operator()
      (const shared_ptr<Record> rec)
    {
      if ( rec->getKind() == d_recKind )
        return false;                       // meaning please copy!
      else
        return true;
    }

  private:

    xeona::RecordKind    d_recKind;

  };

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS           : RecordSet
// ----------------------------------------------------------
//  Description  : holds a set of Records
//  Role         : provide a single point of entry
// ----------------------------------------------------------

//  STATIC DEFINITIONS

logga::spLogger
RecordSet::s_logger = logga::ptrLogStream(); // bind logger on definition

// CREATORS

RecordSet::RecordSet() :
  d_status(e_empty),
  d_records()                               // empty vector
{
```

```
  s_logger->repx(logga::dbug, "constructor call", "");
}

RecordSet::~RecordSet()
{
  s_logger->repx(logga::dbug, "destructor call", "");
}

// MANIPULATORS -- for loading data and such

void RecordSet::setStatus(RecordSet::Status status)  { d_status = status;           }

void RecordSet::addRecord(shared_ptr<Record> record)
  throw(std::exception, xeona::xem_data_issue)    // exception specification
{
  // check for duplicate enabled record identifier and throw if encountered
  const std::string identifier = record->getIdentifier();
  if ( record->getKind() != xeona::e_note    // duplicate notes allowed
       &&
       record->getEnabled()                  // CAUTION: tribool is 'true'
       &&                                    // .. meaning skip disabled records
       existsEnabledRecord(identifier) )     // tests 'true' if already exists
    {
      s_logger->repx(logga::warn, "duplicate record encountered", identifier);
      if ( xeona::nopro == false )           // meaning option '--krazy' not applied
        {
          s_logger->repx(logga::dbug, "will throw xeona::xem_data_issue", "");
          throw xeona::xem_data_issue("duplicate record encountered",
                                      "(record identifier)",
                                      identifier);
        }
    }

  // perform update
  d_records.push_back(record);
}

void
RecordSet::processRecords()
{
  BOOST_FOREACH( shared_ptr<Record> r, d_records )
    r->processFields();
}

// BOUND SUBSETS

std::vector<shared_ptr<Record> >             // may be an empty vector
RecordSet::copySet()
{
  typedef std::vector<shared_ptr<Record> > vpr_type;   // for convenience

  vpr_type fullset(d_records.size());        // CAUTION: must be of sufficient size
  vpr_type::iterator pos;                    // CAUTION: cannot be const_iterator

  pos = std::copy                            // from <algorithm>, 'pos' is not used here
    (d_records.begin(),                      // source container
     d_records.end(),
     fullset.begin());                       // destination container

  return fullset;                            // could be an empty vector
}

std::vector<shared_ptr<Record> >             // may be an empty vector
RecordSet::copySubset
(const xeona::RecordKind recKind)
{
  // CAUTION: note the "copy if false" logic for the
  // 'std::remove_copy_if' predicate

  typedef std::vector<shared_ptr<Record> > vpr_type;   // for convenience

  vpr_type subset(d_records.size());         // CAUTION: must be of sufficient size
  vpr_type::iterator pos;                    // CAUTION: cannot be const_iterator

  pos = std::remove_copy_if                  // from <algorithm>
    (d_records.begin(),                      // source container
     d_records.end(),
     subset.begin(),                         // destination container
     ::IfRecKind(recKind));                  // functor as temporary
  subset.erase(pos, subset.end());           // CAUTION: must trim the trailing junk
```

```
  return subset;                              // could be an empty vector
}

// ACCESSORS

RecordSet::Status                       RecordSet::getStatus()  { return d_status;       }
unsigned                                RecordSet::getCount() { return d_records.size(); }
const std::vector<shared_ptr<Record> >& RecordSet::getRecords() { return d_records;      }

const shared_ptr<Record>
RecordSet::locateRecord
(const std::string& recordIdentifier) const
{
  shared_ptr<Record> ret = locateRecordQuietly(recordIdentifier);
  if ( ! ret )
    s_logger->repx(logga::warn, "record identifier not found", recordIdentifier);
  return ret;
}

const shared_ptr<Record>
RecordSet::locateRecordQuietly
(const std::string& recordIdentifier) const
{
  BOOST_FOREACH( shared_ptr<Record> r, d_records )
    if ( r->getIdentifier() == recordIdentifier )
      return r;
  return shared_ptr<Record>::shared_ptr();   // empty smart pointer
}

const bool
RecordSet::existsEnabledRecord
(const std::string& recordIdentifier) const
{
  BOOST_FOREACH( shared_ptr<Record> r, d_records )
    if ( r->getIdentifier() == recordIdentifier
         &&
         r->getEnabled() )
      {
        return true;
      }
  return false;
}

const shared_ptr<Field>
RecordSet::locateRecordAndField
(const std::string& recordIdentifier,
 const std::string& fieldname) const
{
  // CAUTION: the "=" are correct
  if ( shared_ptr<Record>  r = locateRecord(recordIdentifier) )  // implied "this->"
    if ( shared_ptr<Field> f = r->locateField(fieldname) )
      return f;
  s_logger->repx(logga::warn, "problems finding record and field", "");
  return shared_ptr<Field>::shared_ptr();    // empty smart pointer
}

#ifdef _XUTEST // unit testing only

void
RecordSet::dump
(std::string marker)                        // optional marker to insert
{
  std::ostringstream put;
  put << std::boolalpha;
  put << marker << "\n";
  s_logger->putx(logga::dbug, put);

  BOOST_FOREACH( shared_ptr<Record> r, d_records )
    {
      put << r->getIdentifier()               << "\n";
      put << ""                               << "\n";
      put << "  enabled : " << r->getEnabled() << "\n";
      put << "  kind    : " << r->getKind()    << "\n";
      put << ""                               << "\n";

      BOOST_FOREACH( shared_ptr<Field> f, r->d_fields )
        {
          std::string raw(f->getRawStr());
          unsigned size = xeona::consoleWidth - 22 - 3;     // hardcoded in 'common.cc'
          if ( raw.length() > size )
            {
```

```
              raw.resize(size);
              raw += " >";
            }

          put << "  " << std::setw(20) << std::left
              << f->getName()
              << raw << "\n";
        }
      put << "\n";
      s_logger->putx(logga::dbug, put);      // print at record conclusion
    }

  put << marker << "\n";
  s_logger->putx(logga::dbug, put);
}

#endif // _XUTEST

// ------------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ------------------------------------------------------------
//
//  Place all required template instantiations here.  That is,
//  place all required (or potentially required) header-declared
//  class and member function template instantiations at the end
//  of the implementation file.
//
//  (Alternatively, move the relevant definitions to the header
//  file.  This latter approach would require the unnamed
//  namespace functors to be moved there as well -- thereby
//  making the header more complex than it need be.)
//
//  For simple information on template coding strategies, read
//  Cline (2006) section 35.  If this is insufficient for your
//  needs, refer to Lischner (2003 pp195-199) and Dattatri (2002
//  pp456-461).
//
//  Failure to properly instantiate a template will typically
//  result in a link-time error whenever a (constructor or
//  function) call is made.  The linker usually says: "undefined
//  reference to ...".  If no call is made, then the program
//  should compile normally and run without a murmur.
//
//    Cline, Marshall.  2006.  C++ FAQ lite.
//    [www.parashift.com/c++-faq-lite/templates.html]

template void Field::lexValue(shared_ptr<std::vector<int        > >);
template void Field::lexValue(shared_ptr<std::vector<double     > >);
template void Field::lexValue(shared_ptr<std::vector<bool       > >);
template void Field::lexValue(shared_ptr<std::vector<tribool    > >);
template void Field::lexValue(shared_ptr<std::vector<std::string> >);

template wrap<int        > Record::tieSingle(const std::string&);
template wrap<double     > Record::tieSingle(const std::string&);
template wrap<bool       > Record::tieSingle(const std::string&);
template wrap<tribool    > Record::tieSingle(const std::string&);
template wrap<std::string> Record::tieSingle(const std::string&);

template wrap<int        > Record::tieConstant(const int        );
template wrap<double     > Record::tieConstant(const double     );
template wrap<bool       > Record::tieConstant(const bool       );
template wrap<tribool    > Record::tieConstant(const tribool    );
template wrap<std::string> Record::tieConstant(const std::string);

template shared_ptr<std::vector<int        > > Record::tieTimeseries(const std::string&);
template shared_ptr<std::vector<double     > > Record::tieTimeseries(const std::string&);
template shared_ptr<std::vector<bool       > > Record::tieTimeseries(const std::string&);
template shared_ptr<std::vector<tribool    > > Record::tieTimeseries(const std::string&);
template shared_ptr<std::vector<std::string> > Record::tieTimeseries(const std::string&);

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : si3units.cc
//  file-create-date : Thu 17-Dec-2009 13:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : support for engineering format and SI prefixes / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/si3units.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "si3units.h"         // companion header for this file (place first)

#include "../c/util2.h"       // free functions which offer general utilities 2

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <iomanip>            // setw() and family
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings

#include <cmath>              // C-style maths, abs(), ceil(), floor(), sqrt()

//  CODE

namespace xeona
{
  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::getSiPrefixStr
  // ------------------------------------------------------------

  std::string
  getSiPrefixStr
  (xeona::SiPrefix prefix)
  {
    switch ( prefix )
      {
      case xeona::yocto: return "y";
      case xeona::zepto: return "z";
      case xeona::atto:  return "a";
      case xeona::femto: return "f";
      case xeona::pico:  return "p";
      case xeona::nano:  return "n";
      case xeona::micro: return "u";         // as a substitute for \mu
      case xeona::milli: return "m";
      case xeona::none:  return " ";         // space char
      case xeona::kilo:  return "k";
      case xeona::mega:  return "M";
      case xeona::giga:  return "G";
```

```
    case xeona::tera:  return "T";
    case xeona::peta:  return "P";
    case xeona::exe:   return "E";
    case xeona::zetta: return "Z";
    case xeona::yotta: return "Y";

    case xeona::tooSmall:
    case xeona::tooLarge:
      {
        logga::spLogger logger = logga::ptrLogStream();
        logger->repx(logga::warn, "unsupported SI prefix, enum", prefix);
        return "";
      }

    case xeona::notSet:
      {
        logga::spLogger logger = logga::ptrLogStream();
        logger->repx(logga::warn, "SI prefix currently unset, enum", prefix);
        return "";
      }

    default:
      std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
      return "";
    }
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::getEngineeringPrefix
// ----------------------------------------------------------

xeona::SiPrefix
getEngineeringPrefix
(const double quantity,
 const int    shift)                      // note default value
{
  // threshold
  const double closeToZero = 1.0e-15;      // CAUTION: can reset here

  // bind logger
  static logga::spLogger logger = logga::ptrLogStream();

  // preamble
  double absol = std::abs(quantity);       // refer <cmath>

  // special cases
  if ( absol == 0.0 )                      // strictly floating point zero
    {
      return xeona::none;
    }
  else if ( absol < closeToZero )          // close-to-zero rounding
    {
      logger->repx(logga::adhc, "quantity supplied considered zero", quantity);
      return xeona::none;
    }
  else if ( absol == std::numeric_limits<double>::infinity() )      // refer <limits>
    {
      return xeona::none;
    }

  // apply shift such that 'shift' = 1 means 9999.9 maps to
  // itself and not 9.9999 k
  absol *= std::pow(10.0, -shift);

  // recover SI prefix -- note the difference between "<" and
  // "<=", the former blocks 1000 whereas the latter allows it
  xeona::SiPrefix scale = xeona::notSet;

  if ( absol < 1.0e-24 )
    {
      scale = xeona::tooSmall;
      logger->repx(logga::warn, "excessively small quantity supplied", quantity);
    }
  else if ( absol < 1.0e-21 ) scale = xeona::yocto;
  else if ( absol < 1.0e-18 ) scale = xeona::zepto;
  else if ( absol < 1.0e-15 ) scale = xeona::atto;
  else if ( absol < 1.0e-12 ) scale = xeona::femto;
  else if ( absol < 1.0e-09 ) scale = xeona::pico;
  else if ( absol < 1.0e-06 ) scale = xeona::nano;
  else if ( absol < 1.0e-03 ) scale = xeona::micro;
  else if ( absol < 1.0e+00 ) scale = xeona::milli;
```

```
    else if ( absol < 1.0e+03 ) scale = xeona::none;
    else if ( absol < 1.0e+06 ) scale = xeona::kilo;
    else if ( absol < 1.0e+09 ) scale = xeona::mega;
    else if ( absol < 1.0e+12 ) scale = xeona::giga;
    else if ( absol < 1.0e+15 ) scale = xeona::tera;
    else if ( absol < 1.0e+18 ) scale = xeona::peta;
    else if ( absol < 1.0e+21 ) scale = xeona::exe;
    else if ( absol < 1.0e+24 ) scale = xeona::zetta;
    else if ( absol < 1.0e+27 ) scale = xeona::yotta;
    else
      {
        scale = xeona::tooLarge;
        logger->repx(logga::warn, "excessively large quantity supplied", quantity);
      }

    return scale;
  }

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::fmtEngineering
  // ------------------------------------------------------------

  std::string
  fmtEngineering
  (const double   quantity,
   const unsigned decimals)                  // note default value
  {
    const xeona::SiPrefix prefix = xeona::getEngineeringPrefix(quantity);
    return xeona::fmtEngineeringFix(quantity, prefix, decimals);
  }

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::fmtEngineeringFix
  // ------------------------------------------------------------

  std::string
  fmtEngineeringFix
  (double                quantity,
   const xeona::SiPrefix level,
   const unsigned        decimals,
   const double          closeToZero)
  {
    // constants
    const std::string unitSeparator = " ";   // CAUTION: can also be ""
    const std::string zero          = "0";   // CAUTION: can be "0.0"

    // rescale
    const std::string prefixStr = xeona::getSiPrefixStr(level);
    const double requantity     = quantity / std::pow(10.0, level);

    // close-to-zero rounding as appropriate
    if ( xeona::zero )
      {
        if ( std::abs(quantity) < closeToZero ) quantity = 0.0;
      }

    // stringification
    std::ostringstream oss;
    oss << std::fixed << std::setprecision(decimals) << requantity;
    std::string requantityStr = oss.str();

    // return strings
    if ( xeona::isInf(quantity) )                      // plus or minus inf
      {
        return requantityStr + unitSeparator + " ";    // skip prefix string
      }
    if ( xeona::isNan(quantity) )                      // NaN (the sign is meaningless)
      {
        return requantityStr + unitSeparator + " ";    // skip prefix string
      }
    else if ( quantity == 0.0 )                        // exact match
      {
        return zero + unitSeparator + " ";             // string 'zero' is defined above
      }
    else
      {
        return requantityStr + unitSeparator + prefixStr;
      }
  }

} // namespace 'xeona'
```

```
//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : rundev.cc
//  file-create-date : Wed 25-Jul-2007 07:32 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : principal simulation call / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/simcall.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Principal simulation call.

//  LOCAL AND SYSTEM INCLUDES

#include "simcall.h"          // companion header for this file (place first)

#include "../f/gatesreg.h"    // find and register gateways
#include "../c/datio.h"       // model data io
#include "../c/factory.h"     // entity factory
#include "../c/files.h"       // file path processing
#include "../c/recset.h"      // record set support
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/overseer.h"    // top-level overseer entity (singleton)
#include "../c/conex.h"       // create and connect block interfaces
#include "../b/register.h"    // entity sub-class registrations
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <stdexcept>          // standard exception classes, runtime_error()
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/filesystem.hpp>         // path objects, iterators, useful operations
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting

#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;       // three state boolean
using boost::logic::indeterminate; // allows 'indeterminate' and 'indeterminate()'

//  CODE

// ------------------------------------------------------
```

```
//  FREE FUNCTION   : ::interpreteSimKind
// ------------------------------------------------------

namespace
{
  std::string
  interpretSimKind
  (const xeona::SimKind simKind)
  {
    std::string buffer = "(not overwritten)";
    switch ( simKind )
{
 case xeona::e_notSpecified:   buffer = "not specified";                          break;
 case xeona::e_hollowCall:     buffer = "hollow call";                            break;
 case xeona::e_identifyFile:   buffer = "identify model file - no overwrite";     break;
 case xeona::e_parseModel:     buffer = "parse model file - shallow overwrite";   break;
 case xeona::e_invokeFactory:  buffer = "construct objects - deep overwrite";     break;
 case xeona::e_linkAndConnect: buffer = "link and connect - deep overwrite";      break;
 case xeona::e_firstStepRun:   buffer = "first simulation step - deep overwrite"; break;
 case xeona::e_fullRun:        buffer = "full simulation - deep overwrite";       break;
 default: std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
}
    buffer += " ";
    buffer += boost::str(boost::format("(%d)") % simKind);
    return buffer;
  }
} // unnamed namespace

// ------------------------------------------------------
//  FREE FUNCTION   : ::interpreteSimRet
// ------------------------------------------------------

namespace
{
  std::string
  interpretSimRet
  (const xeona::SimRet simRet)
  {
    std::string buffer = "(not overwritten)";
    switch ( simRet )
      {
      case xeona::e_statusNotKnown:   buffer = "status not known";  break;
      case xeona::e_success:          buffer = "success";           break;
      case xeona::e_modelFileFault:   buffer = "model file fault";  break;
      case xeona::e_infeasibility:    buffer = "hit infeasibility"; break;
      case xeona::e_errantSimulation: buffer = "errant simulation"; break;
      case xeona::e_testCodeUsed:     buffer = "test code used";    break;
      case xeona::e_other:            buffer = "other";             break;
      default: std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
      }
    buffer += " ";
    buffer += boost::str(boost::format("(%d)") % simRet);
    return buffer;
  }
} // unnamed namespace

// ------------------------------------------------------
//  FREE FUNCTION   : xeona::simulate
// ------------------------------------------------------
//  Description  : principal simulation call
//  Role         : runs the simulation to the required 'SimKind' "depth"
//  Caller       : 'main'
//  Status       : working
//
//  Design notes
//
//      The services provided by the code within this unit are
//      mostly procedural, which means a "function-driven
//      programming paradigm" is the most appropriate (Alhir
//      1998, p43-45).  Hence this free function named
//      'xeona::simulate'.
//
//      'xeona::simulate' also relies for support on local
//      (prefixed "::" in unnamed namespace) free functions,
//      because there seemed little reason to code and deploy
//      stateless singleton objects solely for their member
//      function calls.
//
//      This function also undertakes some integrity checks and
//      responds accordingly, usually by downgrading the
//      simulation kind.  Unless, that is, 'xeona::nopro' has
```

```
//      been set to 'true' by the '--krazy' command-line option.
//
//      Logger calls are best made from the most appropriate
//      location -- hence there are quite a number 'repx' calls
//      within this function definition.
//
//  References
//
//      Alhir, Sinan Si.  1998.  UML in a nutshell : a desktop
//        quick reference.  O'Reilly and Associates, Sebastopol,
//        California, USA.  ISBN: 1-56592-448-7.
//
// ------------------------------------------------------

xeona::SimRet                                // simulation return enum
simulate
(const std::string&   modelName,             // from command-line or 'main' func default
 const xeona::SimKind simKind)               // from command-line or 'main' func default
{
  // ---------------------------------
  //  set up logging
  // ---------------------------------

  static logga::spLogger logger = logga::ptrLogStream();  // bind logger
  std::ostringstream put;                                 // used for 'logger->putx' calls

  // ---------------------------------
  //  declare some local variables
  // ---------------------------------

  xeona::SimRet  simret;                     // return value for this function
  xeona::SimKind simkind;                     // modifiable version of 'simKind'

  simret  = xeona::e_success;                // set to success, may be overwritten later
  simkind = simKind;                         // assign original value

  // ---------------------------------
  //  report some initial information
  // ---------------------------------

  logger->repx(logga::dbug, "entering free function", "");
  logger->repx(logga::dbug, "simulation model name", modelName);

  std::string msg = "requested simulation kind";
  switch ( simKind )                         // note the different reporting levels
    {
    case xeona::e_notSpecified:   logger->repx(logga::warn, msg, "notSpecified");   break;
    case xeona::e_hollowCall:     logger->repx(logga::dbug, msg, "hollowCall");     break;
    case xeona::e_identifyFile:   logger->repx(logga::dbug, msg, "identifyFile");   break;
    case xeona::e_parseModel:     logger->repx(logga::dbug, msg, "parseModel");     break;
    case xeona::e_invokeFactory:  logger->repx(logga::dbug, msg, "invokeFactory");  break;
    case xeona::e_linkAndConnect: logger->repx(logga::dbug, msg, "linkAndConnect"); break;
    case xeona::e_firstStepRun:   logger->repx(logga::dbug, msg, "firstStepRun");   break;
    case xeona::e_fullRun:        logger->repx(logga::dbug, msg, "fullRun");        break;
    default: std::clog << "** coding error 03 in source file " << __FILE__ << std::endl;
    }

  // ---------------------------------
  //  * identify the XEM file              // the asterisk means a simkind switch
  // ---------------------------------

  if ( simkind >= xeona::e_identifyFile )  // enum given in "simcall.h"
    {
      logger->repx(logga::dbug, "executing nested code for", "identifyFile");

      // ---------------------------------
      //  establish model path
      // ---------------------------------

      // try to establish the model path as a regular file
      // after removing the pre-defined model file extension
      // 'xeona::modelExt' if present -- in particular, note
      // the call to 'xeona::establishModelFile' which confirms
      // the existence of the file or returns an empty path on
      // failure

      std::string modelStub(modelName);      // construct a local copy
      if ( boost::ends_with(modelStub, xeona::modelExt) )
        {
          boost::erase_tail(modelStub, xeona::modelExt.length());
        }
```

```
boost::filesystem::path model = xeona::establishModelFile(modelStub);

if ( model.empty() )                    // no path established
  {
    logger->repx(logga::warn, "empty model path", model.empty());
    logger->repx(logga::dbug, "simulation downgraded to", "e_identifyFile");
    simkind = xeona::e_identifyFile;    // downgrade simulation kind
    simret  = xeona::e_modelFileFault;
  }
else
  {
    logger->repx(logga::dbug, "model path (complete)", model);  // streamable
    logger->repx(logga::dbug, "model path (leaf)", model.filename());
  }

// ---------------------------------
//  * parse model
// ---------------------------------

if ( simkind >= xeona::e_parseModel )
  {
    logger->repx(logga::dbug, "executing nested code for", "parseModel");

    // ---------------------------------
    //  make record set and io object
    // ---------------------------------

    RecordSet recset;                   // create empty record set
    DataIo dataIo(recset);              // create data input/output admin object

    // ---------------------------------
    //  read and process model
    // ---------------------------------

    put << "   ++ about to read in model" << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);

    // read in model, 'd_rawStr' is loaded but not processed
    logger->repx(logga::dbug, "about to call readModel", "");
    dataIo.readModel(model);

    put << "   ++ about to undertake post-read processing" << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);

    // confirm that model format and svn revision align
    dataIo.confirmModelSvnAlign();

    // locate horizon information -- necessary for the
    // 'processModel' call and ALSO for the production of
    // the TimeHorizon entity 'time-horizon'
    dataIo.locateModelHorizon();        // updates 'Entity::s_horizonSteps'

    const int horizonSteps = Entity::getHorizonSteps();
    put << "  horizon steps : " << horizonSteps << "\n";
    logger->putx(logga::xtra, put);
    logger->addSmartBlank(logga::xtra);

    // process 'd_rawStr' to yield 'd_splitStr' string vectors
    logger->repx(logga::dbug, "about to call processModel", "");
    dataIo.processModel();

    // defensive programming if horizon steps less than two
    if ( ! horizonSteps >= 2 )
      {
        logger->repx(logga::warn, "time-horizon.steps not 2 or more", horizonSteps);
        if ( xeona::nopro )
          {
            logger->repx(logga::info, "defensive coding omitted", "");
            simret  = xeona::e_modelFileFault;
          }
        else
          {
            logger->repx(logga::dbug, "simulation downgraded to", "e_parseModel");
            simkind = xeona::e_parseModel;  // downgrade simulation kind
            simret  = xeona::e_modelFileFault;
          }
      }
```

```
          // ---------------------------------
          //  * invoke entity factory
          // ---------------------------------

          if ( simkind >= xeona::e_invokeFactory )
            {
              logger->repx(logga::dbug, "executing nested code for", "invokeFactory");

              // ---------------------------------
              //  register creators en-masse
              // ---------------------------------

              put << "   ++ "
                  << "about to register entity creators"
                  << "\n";
              logger->addSmartBlank(logga::dbug);
              logger->putx(logga::dbug, put);
              logger->addSmartBlank(logga::dbug);

              logger->repx(logga::dbug, "about to call registerEntyCreators", "");
              xeona::registerEntyCreators(); // one-off call, protected by a warning

              // ---------------------------------
              //  set mandatory entities
              // ---------------------------------

              // the functionality here is duplicated later on
              // (search on "mandatories") when a list of
              // mandatory identifiers is confirmed -- the code
              // here is considerably more complicated and
              // should probably be ripped out at some point

              put << "   ++ about to set the mandatory entities" << "\n";
              logger->addSmartBlank(logga::dbug);
              logger->putx(logga::dbug, put);
              logger->addSmartBlank(logga::dbug);

              // mandatory entity types listed here
              Entity::addWant("TimeHorizon");
              Entity::addWant("Overseer");

# if 0 // 0 = skip code (safe option), 1 = induce fake complaint if not a release build
              if ( xeona::releaseStatus == false )
                {
                  Entity::addWant("FakeForTesting");
                }
# endif // 0

              // ---------------------------------
              //  formulate the production list
              // ---------------------------------

              put << "   ++ about to formulate the production list" << "\n";
              logger->addSmartBlank(logga::dbug);
              logger->putx(logga::dbug, put);
              logger->addSmartBlank(logga::dbug);

              // process entity instances

              typedef std::vector<shared_ptr<Record> > RexSet;   // note also 'RecordSet'

              const RexSet entities = dataIo.getSubset(xeona::e_entity);   // "entity."

              RexSet productionList;
              xeona::tailCombine(productionList, entities);

              put << "    production breakdown"                     << "\n";
              put << "    entity.*  entities : " << entities.size() << "\n";
              logger->putx(logga::xtra, put);
              logger->addSmartBlank(logga::xtra);

              // ---------------------------------
              //  undertake object production
              // ---------------------------------

              put << "   ++ about to start production" << "\n";
              logger->addSmartBlank(logga::dbug);
              logger->putx(logga::dbug, put);
              logger->addSmartBlank(logga::dbug);
```

```
            // the following vector should also remain
            // identical to 'Entity::s_censusFull'

            std::vector<shared_ptr<Entity> > xemFull;     // xem-sourced register

            const std::string tag = "FAC-";     // lead tag

            unsigned facLoop      = 0;
            unsigned xemFullCount = 0;

            BOOST_FOREACH( shared_ptr<Record> r, productionList )
              {
                const std::string id       = r->getIdentifier(); // uniqueness is tested
                const tribool enabled      = r->getEnabled();  // 'non-true' is disabled
                const std::string eTypeStr = r->locateClass(); // empty means failure

                if ( ! enabled )
                  {
                    logger->repx(logga::info, "skipping disabled record", id);
                    continue;
                  }

                const std::string separator = "  ";  // separator
                std::ostringstream put;               // new for each loop
                std::ostringstream oss;               // new for each loop
                oss << "    ";
                oss << tag << std::setw(2) << std::setfill('0') << ++facLoop << "  ";
                oss << eTypeStr
                    << separator
                    << id
                    << "\n";
                logger->addSmartBlank(logga::xtra);
                logger->putx(logga::xtra, oss);
                logger->addSmartBlank(logga::xtra);

                try
                  {
// main production call
xemFull.push_back(EntityFactory::iface()->createEntityBind(eTypeStr, id, *r));

                    // some housekeeping
                    ++xemFullCount;
                    if ( xemFull.size() != xemFullCount )      // defensive programming
                      {
                        logger->repx(logga::warn, "xemFull.size != xemFullCount", "");
                      }
                  }
                catch( const std::out_of_range& e )
                  {
logger->repx(logga::warn, "'std::out_of_range' caught", "details below");
const std::string what = e.what();         // CAUTION: cannot string == directly
put   << "    event : std::out_of_range exception caught"               << "\n";
put   << "    what  : " << what                                        << "\n";
if ( what == "vector::_M_range_check" )    // also occurs with NDEBUG set
  {
    put << "    check : faulty 'std::vector<>::at' call within an entity?" << "\n";
  }
put   << "    next  : will rethrow, application should exit cleanly"     << "\n";
logger->putx(logga::warn, put);
throw;                                     // rethrow
                  }
                catch( const std::exception& e )
                  {
                    logger->repx(logga::warn, "std::exception caught, rethrown", "");
                    throw;                 // rethrow
                  }
              } // boost_foreach production list loop

            // ---------------------------------
            //  set deep write requirement
            // ---------------------------------

            // inform the data io object that the model is
            // now bound -- this will provoke a "deep"
            // (rather than "shallow") write later on

            logger->repx(logga::xtra, "about to call noteModelIsBound", "");
            dataIo.noteModelIsBound(); // 'DataIo::d_records' now 'RecordSet::e_bound'

            // ---------------------------------
            //  conduct post-production checks
```

```
// ---------------------------------

put << "   ++ about to conduct post-production checks" << "\n";
logger->addSmartBlank(logga::dbug);
logger->putx(logga::dbug, put);
logger->addSmartBlank(logga::dbug);

put << "    horizon steps (as used)    : "
    << Entity::getHorizonSteps()
    << "\n"
    << "    horizon interval (as used) : "
    << Entity::getHorizonInterval()
    << "\n";
logger->addSmartBlank(logga::dbug);
logger->putx(logga::dbug, put);
logger->addSmartBlank(logga::dbug);

put << "    xem-sourced entity count       : " << xemFullCount   << "\n";
put << "    xem-sourced entity vector size : " << xemFull.size() << "\n";
logger->addSmartBlank(logga::dbug);
logger->putx(logga::dbug, put);
logger->addSmartBlank(logga::dbug);

// confirm mandatory types

std::vector<std::string> unfulfilled;
Entity::checkWants(unfulfilled);
if ( unfulfilled.empty() )
  {
    logger->repx(logga::dbug, "all mandatory entity types present", "");
  }
else
  {
    int size = unfulfilled.size();
    logger->repx(logga::warn, "unfulfilled mandatory entity types", size);
    put << "    " << "list of unfulfilled mandatory entity types:" << "\n"
        << "\n";
    BOOST_FOREACH( std::string s, unfulfilled )
      put << "      " << s << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);
  }

// ---------------------------------
//  list mandatory objects
// ---------------------------------

// this is a good point to test for mandatory
// entity ids (noting that "time-horizon" which
// has already been found) here and drop back if
// any are absent
//
// the identifier strings themselves are
// hardcoded in 'common.cc'

std::vector<std::string> mandatories;
mandatories.push_back(xeona::timehorizon);    // 'TimeHorizon' entity type
mandatories.push_back(xeona::overseer);       // 'Overseer' entity type

// ---------------------------------
//  confirm mandatory objects
// ---------------------------------

int missingCount = 0;
BOOST_FOREACH( std::string s, mandatories )
  {
    if ( Entity::confirmIdentifier(s) == false )
      {
        put << "      " << s << "\n";
        ++missingCount;
      }
  }
if ( missingCount == 0 )
  {
    logger->repx(logga::dbug, "all mandatory entities found", "");
  }
else
  {
    logger->repx(logga::warn, "mandatory entities not found", missingCount);
    std::ostringstream oss;
```

```
    oss << "    " << "list of mandatory entities not found" << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, oss);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);

    if ( xeona::nopro )
      {
        logger->repx(logga::info, "defensive coding omitted", "");
        simret  = xeona::e_modelFileFault;
      }
    else
      {
        logger->repx(logga::dbug,
                     "simulation downgraded to",
                     "e_invokeFactory");
        simkind = xeona::e_invokeFactory;     // downgrade simulation kind
        simret  = xeona::e_modelFileFault;
      }

  } // missingCount conditional

// ---------------------------------
//  * link and connect
// ---------------------------------

if ( simkind >= xeona::e_linkAndConnect  )
  {
    logger->repx(logga::dbug,
                 "executing nested code for",
                 "link and connect");

    // ---------------------------------
    //  link entities via embedded pointers
    // ---------------------------------

    put << "   ++ about to link entities (information flows)" << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);

    bool linkRet;
    linkRet = Entity::linkAll();    // repeat call never problematic

    logger->repx(logga::dbug, "Entity::linkAll return", linkRet);

    // ---------------------------------
    //  connect cables to sockets
    // ---------------------------------

    put << "   ++ about to connect interfaces (cables to sockets)" << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);

    bool connectRet;
    connectRet = Interface::connectAll();  // repeat call can be problematic

    // a repeat call to 'Interface::connectAll' may
    // be problematic depending on the setting of two
    // hash-conditional in 'connec.cc' -- at the time
    // of writing, a repeat call would be okay

    logger->repx(logga::dbug, "Interface::connectAll return", connectRet);

    // ---------------------------------
    //  register gateways
    // ---------------------------------

    put << "   ++ about to locate gateways and complete their registration"
        << "\n";
    logger->addSmartBlank(logga::dbug);
    logger->putx(logga::dbug, put);
    logger->addSmartBlank(logga::dbug);

    bool registerGatesRet;
    registerGatesRet = xeona::registerGates();

    logger->repx(logga::dbug,
                 "xeona::registerGates return",
                 registerGatesRet);
```

```
        // ---------------------------------
        //  * run simulation
        // ---------------------------------

        if ( simkind >= xeona::e_firstStepRun )
          {
            logger->repx(logga::dbug,
                         "executing nested code for",
                         "run (first or full)");

            unsigned span = 0;         // initialize with nonsensical value

            // ---------------------------------
            //  set steps
            // ---------------------------------

            if ( simKind == xeona::e_firstStepRun )
              {
                span = 1;              // just one step

                put << "   ++ about to simulate just one step" << "\n";
                logger->addSmartBlank(logga::dbug);
                logger->putx(logga::dbug, put);
                logger->addSmartBlank(logga::dbug);
              }
            else if (simkind == xeona::e_fullRun )
              {
                span = Entity::getHorizonSteps();  // XEM file value, 2 or more

                put << "   ++ about to simulate the entire horizon using span: "
                    << span << "\n";
                logger->addSmartBlank(logga::dbug);
                logger->putx(logga::dbug, put);
                logger->addSmartBlank(logga::dbug);
              }
            else
              {
                std::clog << "** coding error 04 in source file " << __FILE__
                          << std::endl;
              }

            // ---------------------------------
            //  invoke 'Overseer::run'
            // ---------------------------------

            // the 'Overseer' identifier "overseer" is hardcoded
            // in 'common.cc'

            // CAUTION: the following "downcast" is needed
            // because 'Entity' and 'FullEntity' do not
            // possess 'run' calls -- note also a shared
            // pointer 'dynamic_pointer_cast' is required
            // instead of the more normal raw pointer
            // 'dynamic_cast.'

            shared_ptr<Overseer> overseer = Entity::retOverseer();

            bool runRet = overseer->run(span);   // WOW: run the simulation!

            if ( runRet == true )
              {
                logger->repx(logga::info, "Overseer::run call returned",runRet);
              }
            else
              {
                logger->repx(logga::warn, "Overseer::run call returned",runRet);
                simret = xeona::e_infeasibility;  // assume infeasibility
              }

          } // 'xeona::e_firstStepRun' nesting, maybe also 'xeona::e_fullRun'

      } // 'xeona::e_linkAndConnect' nesting

  } // xeona::e_bindModel nesting

// CAUTION: note that entities are not destructed
// until after the 'main' function returns

// ---------------------------------
// model last-run information
```

```
          // ---------------------------------

          put << "   ++ about to update model last-run information" << "\n";
          logger->addSmartBlank(logga::dbug);
          logger->putx(logga::dbug, put);
          logger->addSmartBlank(logga::dbug);

          const std::string ret
            = xeona::modelStringDelim
            + ::interpretSimRet(simret)
            + xeona::modelStringDelim;
          const std::string kind
            = xeona::modelStringDelim
            + ::interpretSimKind(simkind)
            + xeona::modelStringDelim;

          put << "    model last run information" << "\n"
              << "    return : " << ret           << "\n"
              << "    kind   : " << kind          << "\n";
          logger->putx(logga::xtra, put);
          logger->addSmartBlank(logga::xtra);

          dataIo.updateModelRunTime(ret, kind);

          // ---------------------------------
          // write model
          // ---------------------------------

          // the behavior of 'writeModel' depends on whether
          // 'DataIo::noteModelIsBound' has been called or not
          // -- if not then a shallow write will result,
          // otherwise a deep write will be used
          //
          // a shallow write can be enforced through the use of
          // "--mode 3" on the command-line (but no simulation
          // will be run)

          put << "   ++ about to write out model" << "\n";
          logger->addSmartBlank(logga::dbug);
          logger->putx(logga::dbug, put);
          logger->addSmartBlank(logga::dbug);

          dataIo.writeModel(model);          // takes a 'boost::filesystem::path' object

        } // xeona::e_parseModel nesting

    } // xeona::e_identifyFile nesting

  // ---------------------------------
  //  end of function return
  // ---------------------------------

  logger->repx(logga::xtra, "leaving free function, returning", simret);
  return simret;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : stats.cc
//  file-create-date : Wed 16-Sep-2009 13:34 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : on-the-fly statistical calculations / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/stats.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  File intentionally blank.

//  LOCAL AND SYSTEM INCLUDES

#include "stats.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// CAUTION: intentionally blank, all code is in the header

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : tsops.cc
//  file-create-date : Wed 16-Sep-2009 16:57 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : overloaded operators for timeseries / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/tsops.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "tsops.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::vectorSum <>
// ------------------------------------------------------------

namespace xeona
{
  template <typename T>
  T
  vectorSum
  (const shared_ptr<std::vector<T> >& one)
  {
    // logging support
    static logga::spLogger logger = logga::ptrLogStream();

    // integrity checks
    if ( ! one )
      {
        logger->repx(logga::warn, "supplied vector empty/null", "");
        return T();
      }
    const int len = one->size();

    // active code
    T buf(0);                                 // default construction [1]
    for ( int i = 0; i < len; ++i )
      {
        buf += one->at(i);
      }
```

```
    // return
    return buf;

    // CAUTION: [1] the statement "T buf;" will not work.
  }

} // namespace 'xeona'

// ------------------------------------------------------
//  FREE FUNCTION   : operator+ <>
// ------------------------------------------------------

template <typename T>
shared_ptr<std::vector<T> >
operator+
(const shared_ptr<std::vector<T> >& one,
 const shared_ptr<std::vector<T> >& two)
{
  // logging support
  static logga::spLogger logger = logga::ptrLogStream();

  // integrity checks
  if ( ! one || ! two )
    {
      logger->repx(logga::warn, "supplied vector(s) empty/null", "");
      return shared_ptr<std::vector<T> >();
    }
  const int lenOne = one->size();
  const int lenTwo = two->size();
  if ( lenOne != lenTwo )
    {
      std::ostringstream oss;
      oss << lenOne << " : " << lenTwo;
      logger->repx(logga::warn, "vector length mismatch, one : two", oss.str());
      return shared_ptr<std::vector<T> >();
    }

  // active code
  shared_ptr<std::vector<T> > buf(new std::vector<T>());
  for ( int i = 0; i < lenOne; ++i )
    {
      buf->push_back(one->at(i) + two->at(i));    // note the underlying operator is here
    }

  // return
  return buf;
}

// ------------------------------------------------------
//  FREE FUNCTION   : operator- <>
// ------------------------------------------------------

template <typename T>
shared_ptr<std::vector<T> >
operator-
(const shared_ptr<std::vector<T> >& one,
 const shared_ptr<std::vector<T> >& two)
{
  // logging support
  static logga::spLogger logger = logga::ptrLogStream();

  // integrity checks
  if ( ! one || ! two )
    {
      logger->repx(logga::warn, "supplied vector(s) empty/null", "");
      return shared_ptr<std::vector<T> >();
    }
  const int lenOne = one->size();
  const int lenTwo = two->size();
  if ( lenOne != lenTwo )
    {
      std::ostringstream oss;
      oss << lenOne << " : " << lenTwo;
      logger->repx(logga::warn, "vector length mismatch, one : two", oss.str());
      return shared_ptr<std::vector<T> >();
    }

  // active code
  shared_ptr<std::vector<T> > buf(new std::vector<T>());
  for ( int i = 0; i < lenOne; ++i )
    {
```

```
      buf->push_back(one->at(i) - two->at(i));    // note the underlying operator is here
    }

  // return
  return buf;
}

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------

template int    xeona::vectorSum(const shared_ptr<std::vector<int   > >&);
template double xeona::vectorSum(const shared_ptr<std::vector<double> >&);

template shared_ptr<std::vector<int   > > operator+
(const shared_ptr<std::vector<int   > >&,
 const shared_ptr<std::vector<int   > >&);
template shared_ptr<std::vector<double> > operator+
(const shared_ptr<std::vector<double> >&,
 const shared_ptr<std::vector<double> >&);

template shared_ptr<std::vector<int   > > operator-
(const shared_ptr<std::vector<int   > >&,
 const shared_ptr<std::vector<int   > >&);
template shared_ptr<std::vector<double> > operator-
(const shared_ptr<std::vector<double> >&,
 const shared_ptr<std::vector<double> >&);

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : utils.cc
//  file-create-date : Thu 14-Jun-2007 15:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions which offer general utilities 1 / implementation
//  file-status      : ongoing
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/util1.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "util1.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <fstream>            // file-based io
#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <iterator>           // STL additional iterators, std::distance()
#include <string>             // C++ strings

#include <cctype>             // C-style char classification, case conversion
#include <cmath>              // C-style maths, ceil(), floor(), sqrt()
#include <cstdlib>            // C-style exit(), getenv(), system(), NULL, EXIT_SUCCESS
#include <ctime>              // C-style time and date functions

#include <boost/random/mersenne_twister.hpp>    // random engine
#include <boost/random/uniform_int.hpp>         // random distribution
#include <boost/random/uniform_real.hpp>        // random distribution
#include <boost/random/variate_generator.hpp>   // random generator

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/cast.hpp>               // numeric_cast<>()
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting

//  FORWARD (PARTIAL) DECLARATIONS

class Record;                                 // for 'tailCombine'
class Block;                                  // for 'mapCombine'
class Class;                                  // for 'mapCombine'
class DomainController;                       // for 'mapCombine'
class Entity;                                 // for 'mapCombine'
class Gateway;                                // for 'mapCombine'

//  CODE

namespace xeona
{

  // ---------------------------------------------------------
```

```
//  FREE FUNCTION   : xeona::getRandom (int, int)
// ------------------------------------------------------
//  Description  : retrieve random ints in required range
//  Status       : preliminary testing complete
//
//  Design notes
//
//      The Boost.Random library has been accepted into the TR1
//      but is not explicitly supported by g++ 4.1.  This code
//      relies on the Ubuntu package libboost-dev (Boost C++
//      Libraries development files) for the necessary
//      Boost.Random headers.  That said, explicit linking to
//      say libboost_random is not required.
//
//      Reseeding
//
//      The global variable 'xeona::pepper', if true, creates a
//      different seed for every run.  Otherwise the random
//      engine default is used which leads to repeatability
//      between runs.
//
//  CAUTIONS
//
//      See issues highlighted in the code proper.
//
//  See also
//
//      Boost.Random documentation.
//
//      Becker (2007 ch 13) devotes a chapter to the
//      Boost.Random library.  But note that Karlsson (2006)
//      does *not* cover the library at all.
//
//  References
//
//      Becker, Pete.  2007.  The C++ Standard Library
//        extensions : a tutorial and reference.  Addison-Wesley,
//        Upper Saddle River, New Jersey, USA.  ISBN
//        0-321-41299-0.
//
// ------------------------------------------------------

int                                     // retrieve a random number
getRandom
(int lower,                             // inclusive lower bound
 int upper)                             // inclusive upper bound
{
  // logger disabled due to high number of calls
  static logga::spLogger logger = logga::ptrLogStream();
  //std::string str = boost::str(boost::format("%1%, %2%") % lower % upper);
  //logger->repx(logga::dbug, "arguments lower, upper", str);

  // select a predefined pseudo-random number engine, in
  // this case based on the 'mersenne_twister' algorithm
  static boost::mt19937 eng;            // CAUTION: use static [1], do not seed [2]

  // create a variate_generator object
  boost::variate_generator              // CAUTION: omit static [3]
    <
    boost::mt19937&,                    // CAUTION: ampersand required [4]
    boost::uniform_int<>                // select uniform integer distribution
    >
    rint                                // variate_generator object name
    (eng,                               // bind generator
     boost::uniform_int<>(lower, upper));  // .. and distribution

  // [1] the engine should persist between function calls
  // [2] the engine should not be reseeded on each function call
  // [3] static will fix the range lower, upper on first call, not what
  //     is wanted (although static should be faster)
  // [4] see Becker (2007 p337)
  // [5] std::time(0) is not a very good random seed -- when
  //     called in rapid succession, it will return the same value
  //     and hence the same random number sequences will ensue
  // [6] to ensure run to run repeatability

  static bool firstCall = true;                 // note first call status
  if ( firstCall )
    {
      if ( xeona::pepper == true )              // controls which seed to use
        {
```

```
            std::time_t stamp = std::time(0);     // CAUTION: std::time is not ideal [5]
            boost::mt19937::result_type seed      // better than hardcoding 'unsigned'
              = static_cast<boost::mt19937::result_type>(stamp);
            eng.seed(seed);
            firstCall = false;
            logger->repx(logga::info, "integer form first call, time seed", seed);
          }
        else
          {
            firstCall = false;
            logger->repx(logga::info, "integer first call, default seed", "");
          }
      }

    return rint();                                // return random number
  }

  // ----------------------------------------------------------
  //  FREE FUNCTION   : xeona::getRandom (double, double)
  // ----------------------------------------------------------
  //  Description  : retrieve random doubles in required range
  //  Status       : preliminary testing complete
  //
  //  Design notes
  //
  //      See getRandom (int, int).
  //
  // ----------------------------------------------------------

  double                                      // retrieve a random number
  getRandom
  (double lower,                              // inclusive lower bound
   double upper)                              // inclusive upper bound
  {
    // logger disabled due to high number of calls
    static logga::spLogger logger = logga::ptrLogStream();
    //std::string str = boost::str(boost::format("%1%, %2%") % lower % upper);
    //logger->repx(logga::dbug, "arguments lower, upper", str);

    // select a predefined pseudo-random number engine, in
    // this case based on the 'mersenne_twister' algorithm
    static boost::mt19937 eng;                // CAUTION: use static [1], do not seed [2]

    // create a variate_generator object
    boost::variate_generator                  // CAUTION: omit static [3]
      <
      boost::mt19937&,                        // CAUTION: ampersand required [4]
      boost::uniform_real<>                   // select uniform real distribution
      >
      rdouble                                 // variate_generator object name
      (eng,                                   // bind generator
       boost::uniform_real<>(lower, upper)); // .. and distribution

    // [1] the engine should persist between function calls
    // [2] the engine should not be reseeded on each function call
    // [3] static will fix the range lower, upper on first call, not what
    //     is wanted (although static should be faster)
    // [4] see Becker (2007 p337)
    // [5] std::time(0) is not a very good random seed -- when
    //     called in rapid succession, it will return the same value
    //     and hence the same random number sequences will ensue
    // [6] to ensure run to run repeatability

    static bool firstCall = true;                 // note first call status
    if ( firstCall )
      {
        if ( xeona::pepper == true )              // controls which seed to use
          {
            std::time_t stamp = std::time(0);     // CAUTION: std::time is not ideal [5]
            boost::mt19937::result_type seed      // better than hardcoding 'unsigned'
              = static_cast<boost::mt19937::result_type>(stamp);
            eng.seed(seed);
            firstCall = false;
            logger->repx(logga::info, "real form first call, time seed", seed);
          }
        else
          {
            eng.seed();                          // use default internal state [6]
            firstCall = false;
            logger->repx(logga::info, "real first call, default seed", "");
          }
```

```
    }

  return rdouble();                             // return random number
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::fmtQuantity
// ------------------------------------------------------------
//  Purpose      : return a quantity in resolved SI format
//  Status       : working
//
//  Design notes
//
//      Format %d for (double) 1000000 -> 1.0e+06.
//      Format %d for    (int) 1000000 -> 1000000.
//
//      The SI micro prefix 'u' is a stand-in for 'mu' (unicode
//      U+00B5, ascii dec 181 or octal 265 , LaTeX \mu).  One
//      can use char '\256' but this needs extended-ASCII
//      support on your (bash) shell -- to check if this is
//      possible, try:
//
//          $ printf "%b\n" "\256" # (R) registered trademark symbol
//
//  Examples using 'W'
//
//          200      ->  0.2 kW
//          3.3e+05  ->  330 kW
//          -4.4e+06 -> -4.4 MW
//
//  Note on 'tonne' unit
//
//      The base SI mass unit is 'kg'.  Hence clients passing
//      't' must do their own local conversion first:
//
//          double mass = 300;              // 300 kg
//          fmtQuantity(mass/1000, "t");    // 0.3 t
//
//  Limitations (which arise from the SI system)
//
//          1000 kg  ->  1 kkg (t)
//          0.001 kg ->  1 mkg (g)
//          0.1 t    ->  100 mt (kg)
//
// ------------------------------------------------------------

std::string
fmtQuatity
(const int         value,                   // wrapper for double version
 const std::string baseUnit)
{
  return fmtQuantity(boost::numeric_cast<double>(value), baseUnit);
}

std::string
fmtQuantity
(const double      value,                   // numerical value
 const std::string baseUnit)                // for instance, 'W' for Watts
{
  std::string pre;                          // SI prefix
  std::string pmv;                          // prefix-modified value
  std::string unt;                          // new unit
  std::string buf;                          // buffer to return

  if ( value == 0.0 )                       // strictly zero in floating point arithmetic
    {
      pre = "";                             // generally no prefix
      if ( baseUnit == "W" ) pre = "k";     // kW is an exception
      pmv = boost::str(boost::format("%' 'd") % 0);
    }
  else if ( std::abs(value) < 1.0e-06 )     // below one micro-baseUnit
    {
      pre = "n";                            // 'nano'
      pmv = boost::str(boost::format("%' 'd") % (value * 1.0e+09));
    }
  else if ( std::abs(value) < 1.0e-03 )     // below one milli-baseUnit
    {
      pre = "u";                            // 'micro' stand-in
   // pre = "\265";                         // 'micro' in extended ASCII (see note above)
      pmv = boost::str(boost::format("%' 'd") % (value * 1.0e+06));
    }
```

```
    else if ( std::abs(value) < 1.0e+00 )    // below one baseUnit
      {
        pre = "m";                           // 'milli'
        pmv = boost::str(boost::format("%' 'd") % (value * 1.0e+03));
      }
    else if ( std::abs(value) < 1.0e+03 )    // below one kilo-baseUnit
      {
        pre = "";                            // nothing
        pmv = boost::str(boost::format("%' 'd") % value);
      }
    else if ( std::abs(value) < 1.0e+06 )    // below one mega-baseUnit
      {
        pre = "k";                           // 'kilo'
        pmv = boost::str(boost::format("%' 'd") % (value / 1.0e+03));
      }
    else if ( std::abs(value) < 1.0e+09 )    // below one giga-baseUnit
      {
        pre = "M";                           // 'mega'
        pmv = boost::str(boost::format("%' 'd") % (value / 1.0e+06));
      }
    else if ( std::abs(value) < 1.0e+12 )    // below one tera-baseUnit
      {
        pre = "G";                           // 'giga'
        pmv = boost::str(boost::format("%' 'd") % (value / 1.0e+09));
      }
    else if ( std::abs(value) < 1.0e+15 )    // below one peta-baseUnit
      {
        pre = "T";                           // 'tera'
        pmv = boost::str(boost::format("%' 'd") % (value / 1.0e+12));
      }
    else                                     // remaining
      {
        pre = "P";                           // 'peta'
        pmv = boost::str(boost::format("%' 'd") % (value / 1.0e+15));
      }

    unt = pre + baseUnit;

    // for some units of mass, append a useful equivalent in parentheses

    if ( unt == "ukg" )   unt += " (mg)";    // micro-kg, the 'u' version
    if ( unt == "\256kg") unt += " (mg)";    // micro-kg, the 'mu' version
    if ( unt == "mkg" )   unt += " (g)";     // milli-kg
    if ( unt == "kkg" )   unt += " (t)";     // kilo-kg
    if ( unt == "Mkg" )   unt += " (kt)";    // mega-kg
    if ( unt == "Gkg" )   unt += " (Mt)";    // giga-kg
    if ( unt == "Tkg" )   unt += " (Gt)";    // tera-kg

    if ( unt == "ut" )    unt += " (g)";     // micro-tonne, the 'u' version
    if ( unt == "\256t")  unt += " (mg)";    // micro-tonne, the 'mu' version
    if ( unt == "mt" )    unt += " (kg)";    // milli-tonne

    // create final string

    buf = pmv + " " + unt;
    return buf;
  }

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::fmtPriceRate
  // ------------------------------------------------------------
  //  Purpose      : return a price rate in readable 'kilo' format
  //
  //  Design notes
  //
  //      The functions fmtPriceRate(..) and fmtQuantity(..)
  //      provide more readable output when streaming UnitBid
  //      objects.  If they prove more generally useful, they
  //      could be extended and moved to xeona:: scope.
  //
  //      Regarding the monetary currency unit, "$" is now
  //      hardcoded from commit r3951;
  //
  //  Examples ($ hardcoded)
  //
  //          0.2e-03   W  ->  0.20 $/kW
  //          0.201e-03 W  ->  0.201 $/kW
  //          550       kg ->  550000.00 $/kkg ($/t)

  //
  //  Limitations
```

```
//
//      The 'kilo' prefix is the only one currently supported.
//
// -----------------------------------------------------------

std::string
fmtPriceRate
(const int         priceRate,              // wrapper for double version
 const std::string baseUnit)               // for instance, 'W' for Watts
{
  return fmtPriceRate(boost::numeric_cast<double>(priceRate), baseUnit);
}

std::string
fmtPriceRate
(const double      priceRate,              // numerical value
 const std::string baseUnit)               // for instance, 'W' for Watts
{
  std::string buf;

  std::string unit = "$/k" + baseUnit;
  if ( baseUnit == "kg" ) unit += " ($/t)";  // for "kkg (t)"
  std::string two  = boost::str(boost::format("%' '.2f") % (1000 * priceRate));
  std::string all  = boost::str(boost::format("%' 'd")   % (1000 * priceRate));
  if ( two.length() > all.length() )
    {
      buf = two + " " + unit;              // trailing zero fixed (.2f) form
    }
  else
    {
      buf = all + " " + unit;              // straight decimal (d) form
    }
  return buf;
}

// -----------------------------------------------------------
//  FREE FUNCTION   : xeona::vectorRepeat <>
// -----------------------------------------------------------
//  Description  : create a vector from a repeating pattern
//  Role         : used to complete incomplete timeseries
//  Techniques   : STL 'std::copy'
//  Status       : complete
//
//  Usage
//
//      const int horizon = 7;
//      std::vector<std::string> pattern;
//      std::vector<std::string> timeseries(horizon);
//      pattern.push_back("aa");
//      pattern.push_back("bb");
//      pattern.push_back("cc");
//      xeona::vectorRepeat(pattern, timeseries);
//
//  Design notes
//
//      This function is quite robust.  It works for any value
//      of 'horizon', be it shorter, equal, or longer than the
//      length of the 'pattern' vector and also for zero
//      (although not useful).
//
//      See Lischner (2003 p331) for details on 'std::copy'.
//
// -----------------------------------------------------------

template <typename T>                      // meaning a std::vector<T> container
bool                                       // returns false if pattern is empty
vectorRepeat
(const std::vector<T>& pattern,            // pattern vector, must not be empty
 std::vector<T>&       fullset)            // output vector must be correct length
{
  static logga::spLogger logger = logga::ptrLogStream();
  logger->repx(logga::xtra, "entering member function", "");

  // complain if pattern is empty, because nothing sensible can be done
  if ( pattern.empty() )
    {
      logger->repx(logga::warn, "pattern vector is empty", pattern.size());
      return false;
    }

  // determine output length
```

```
  int horizon = fullset.size();

  // calculate the number of loops required
  int patternLength = pattern.size();      // can be longer than the horizon
  double div                               // CAUTION: use floating point arithmetic
    = static_cast<double>(horizon)
    / static_cast<double>(patternLength);
  double ceiling = std::ceil(div);         // round up to next integer as required
  int loops = static_cast<int>(ceiling);   // 'ceiling' is "integer-valued" anyway

  // prepare the output vector
  fullset.clear();                         // empty the container
  fullset.resize(patternLength * loops);   // ", x" can be useful for testing

  // loop zero or more times as the case may be
  typename std::vector<T>::iterator pos;   // CAUTION: 'typename' essential
  pos = fullset.begin();                   // set fullset iterator to beginning
  for ( int count = 0; count < loops; ++count )
    {
      pos = std::copy                      // CAUTION: must update iterator
        (pattern.begin(),
         pattern.end(),
         pos);
    }

  // truncate if necessary
  fullset.resize(horizon);                 // when horizon < patternLength * loops

  return true;
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::formatDuration
// ----------------------------------------------------------
//
//  CAUTION: fractional seconds (displaying subseconds)
//
//      Fractional seconds depend on the resolution of the
//      system timer and special considerations are needed to
//      write portable code.  This code does not attempt that.
//      See the reference below for the correct strategy.
//
//  References
//
//      The Boost C++ Libraries (BoostBook Subset)
//        > Boost.Date_Time
//          > Posix Time
//            > Introduction
//              > discussion on 'ticks_per_second()'
//
// ----------------------------------------------------------

std::string
formatDuration
(const boost::posix_time::time_duration& delta)
{
  boost::posix_time::time_duration onesec(0, 0, 1);
  if ( delta < onesec )
    {
      return "(under one second)";
    }
  else
    {
      std::ostringstream buf;
      buf << boost::format("%02f")  % delta.hours()  // set to two or three digits
          << boost::format(":%02f") % delta.minutes()
          << boost::format(":%02f") % delta.seconds();
      return buf.str();
    }
}

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::tailCombine <>
// ----------------------------------------------------------
//  Description  : append sequence container 'tail' to 'combined' and thus modify
//  Role         : general, specifically used to combine record subsets
//  Techniques   : explicit template instantiations in source file
//  Headers      : requires <algorithm>, <iterator>
//  Status       : complete
//
//  Design notes
```

```
//
//      The term "combine" was chosen to describe the processes
//      of making one vector from two.  Regarding other
//      definitions, the term "merge" is restricted to sorted
//      sequences and is thus not appropriate.  The term
//      "concatenate" could be used, but is often applied just to
//      strings and that practice is maintained here.  The term
//      "addition" has mathematical meanings.
//
//      For the application in mind, the two sequence containers
//      have no common elements singly or together, that is, no
//      two elements would equate.  This however is not a
//      requirement for the following code to work.
//
//      In most implementations, the 'merge' function template
//      from <algorithm> could work.  Usage by unsorted sequences
//      is not formally supported by the C++ standard and
//      Josuttis (1999 p417) notes that "for unsorted ranges you
//      should call copy() twice, instead of merge(), to be
//      portable".
//
//      The 'copy' function template from <algorithm> is used
//      with the 'back_inserter' function template from
//      <iterator>.  This application is demonstrated in Josuttis
//      (1999 p364) with the comment "use back_inserter to insert
//      instead of overwrite".  For more information on the
//      'back_inserter' see Lischner (2003 p538) and Stephens
//      etal (2006 pp266-267).
//
//      If this function template is used exclusively by vectors,
//      the following line should speed up the code, but is not
//      essential (change the function name too):
//
//          combined.reserve(combined.size() + tail.size());
//
//  CAUTION: supported sequence containers
//
//      tested using     : vector, list, deque
//      not tested using : valarray, string (both pseudo containers anyway)
//
// ------------------------------------------------------------

template <typename S>                       // 'S' is a specialized sequence container
void
tailCombine
(S&       combined,                         // resultant container, need not be empty
 const S& tail)                             // container to be "back inserted"
{
  static logga::spLogger logger = logga::ptrLogStream();
  logger->repx(logga::xtra, "entering free function, tail size", tail.size());

  std::copy(tail.begin(),                   // <algorithm>
            tail.end(),
            std::back_inserter(combined)); // <iterator>
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::mapCombine <>
// ------------------------------------------------------------
//  Description  : merge map 'addition' with map 'combined' and thus modify the latter
//  Role         : general, specifically written for registering gateways
//  Techniques   : explicit template instantiations in source file
//  Caution      : requires UNIQUE keys, warnings issued for duplicates
//  Headers      : requires <map> and probably <utility>
//  Status       : complete
//
//  Design notes
//
//      The implementation use here is not particularly
//      STL-like.  However attempts to use 'std::copy' and
//      'std::inserter' failed, as did attempts to use
//      'std::merge'.
//
//      See Lischner (2003 p604) for code to modify a key in an
//      existing map.
//
//  Reference
//
//      Lischner, Ray.  2003.  C++ in a nutshell : a language and
//        library reference.  O'Reilly and Associates, Sebastopol,
//        California, USA.  ISBN 0-596-00298-X.
```

```
//
// ------------------------------------------------------

template <typename S>
void
mapCombine
(S&       combined,
 const S& addition)
{
  // logging support
  static logga::spLogger logger = logga::ptrLogStream();

  // active code
  const int orig = combined.size();        // just for reporting
  typename S::const_iterator pos;          // CAUTION: 'typename' and 'const_' essential
  for ( pos  = addition.begin();
        pos != addition.end();
        ++pos)
    {
      if ( ! combined.insert(std::make_pair(pos->first, pos->second)).second )
        {
          logger->repx(logga::warn, "map insert failed, key", pos->first);
        }
    }
  const int delta = combined.size() - orig;
  logger->repx(logga::adhc, "number of added elements", delta);
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::trimLeadingDigits
// ------------------------------------------------------------
//  Description  : trims leading digits from given string
//  Role         : various, including use by 'xeona::demangle'
//  Techniques   : C-style 'std::isdigit' from <cctype>
//  Status       : complete
//
//  CAUTION: do not convert 'str' to pass-by-ref
//
//      This function is sometimes passed a char-star and
//      pass-by-ref does not work.
//
// ------------------------------------------------------------

std::string
trimLeadingDigits
(std::string str)
{
  if ( ! str.empty() )                     // protection needed
    while ( std::isdigit(str.at(0)) )      // see <cctype>
      str.erase(0, 1);                     // remove first char
  return str;
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::demangle
// ------------------------------------------------------------
//  Description  : attempts to reinterpret straightforward g++ typenames
//  Role         : repairing typeid names from GCC RTTI
//  Techniques   : simple reverse engineering
//  Status       : complete
//
//  CAUTION: contains g++ compiler specific code
//
//      The C preprocessor will issue a warning if this file is
//      not compiled using GNU g++.
//
//  Design notes
//
//      This function attempts to "demangle" a compiler
//      decorated name.
//
//      Ideally, this function should be a wrapper to a
//      compiler or demangling library API.  However such APIs
//      do not appear to exist for GCC (although they do for
//      other compilers, Hewlett Packard HPUX for example).
//
//      Note also that the 'c++filt' command-line utility does
//      exactly what is required.
//
//      For user-defined classes, g++ typeid(C).name() reports
//      in the form give below.  This code simply strips off
```

```
  //      the leading character count in such cases.
  //
  //          1A
  //          3Now
  //          15MyLongClassName
  //
  //      Note that a leading "P" indicates a raw pointer or (in
  //      my experience) a 'shared_ptr'.  For user-defined types,
  //      this case is dealt with in the code -- however, for
  //      user-defined derived types, only the base class is
  //      revealed by 'typeid'.  The following approach is thus
  //      more informative (assuming member function code):
  //
  //          std::string subtypeName;
  //          subtypeName = xeona::demangle(typeid(*this).name())
  //
  //      Run-time type info is expensive so it may be worth
  //      placing the calling code in a debug only block.
  //
  //      See also my test file: 'frag-rtti-2.cc'.
  //
  //      Fog (2008) describes name mangling in general and GNU
  //      3.* and later name mangling more specifically.
  //
  //  References
  //
  //      Fog, Agner.  2008.  Calling conventions for different
  //        C++ compilers and operating systems.  Copenhagen
  //        University College of Engineering.  Last updated
  //        2008-06-29.  [calling_conventions.pdf].  See in
  //        particular, section 8.5 : Gnu3 name mangling.
  //
  // ----------------------------------------------------------

  std::string                                 // this return may be streamed
  demangle                                    // GCC-specific for fundamental typenames
  (std::string tid)                           // most often 'tid' is a 'const char*'
  {
    // skip if empty string
    if ( tid.empty() ) return "";

    // try some common fundamental types
    if      ( tid == typeid(void        ).name() ) return "void";     // GCC "v"
    else if ( tid == typeid(std::string ).name() ) return "string";   // GCC "Ss"
    else if ( tid == typeid(int         ).name() ) return "int";      // GCC "i"
    else if ( tid == typeid(unsigned    ).name() ) return "unsigned"; // GCC "j"
    else if ( tid == typeid(double      ).name() ) return "double";   // GCC "d"
    else if ( tid == typeid(bool        ).name() ) return "bool";     // GCC "b"
    else if ( tid == typeid(char        ).name() ) return "char";     // GCC "c"
    else if ( tid == typeid(char*       ).name() ) return "char*";    // GCC "Pc"
    else if ( tid == typeid(std::vector<std::string>).name() ) return "vector<string>";
    else if ( tid == typeid(std::vector<int>        ).name() ) return "vector<int>";
    else if ( tid == typeid(std::vector<unsigned>   ).name() ) return "vector<unsigned>";
    else if ( tid == typeid(std::vector<double>     ).name() ) return "vector<double>";
    else if ( tid == typeid(std::vector<bool>       ).name() ) return "vector<bool>";
    else if ( tid == typeid(std::vector<char>       ).name() ) return "vector<char>";

    // then assume, perhaps falsely, a user-defined type

#ifdef __GNUG__                             // CAUTION: g++ compiler specific code block

    else
      {
        if ( tid.substr(0, 1) == "P" )      // probably a raw or shared pointer
          {
            // need to dereference pointer before making 'typeid' call
            logga::spLogger logger = logga::ptrLogStream();
            logger->repx(logga::warn, "dereference prior to 'typeid' call", tid);
            // active code
            const std::string comment = "base class pointer";
            tid.erase(0, 1);                // strip the leading "P"
            return comment + " " + xeona::trimLeadingDigits(tid);
          }
        else if ( tid.substr(tid.length() - 1, 1) == "E" )  // avoid unnecessary effort
          {
            // this code is based on the observation that:
            //
            //     "5Class<Type>" mangles to "5ClassI4TypeE"
            //
            // this code is also specific to 'xeona' although a
            // more general regex pattern match could easily be
```

```
            // written
            //
            // some calls in this block use the 'Boost.Foreach'
            // and 'Boost.String_algo' header-only libraries --
            // which need to be present (or instead factor out
            // the offending calls using the STL library)

            std::vector<std::pair<std::string, std::string> > names;
            names.push_back(std::make_pair("CmOxidize"      , "Oxid"));
            names.push_back(std::make_pair("CmCarbonCert"   , "Cert"));
            names.push_back(std::make_pair("CmCarbonSeq"    , "Cseq"));
            names.push_back(std::make_pair("CmElectricity"  , "Elec"));
            names.push_back(std::make_pair("CmWork"         , "Work"));
            names.push_back(std::make_pair("CmHeat"         , "Heat"));
            names.push_back(std::make_pair("CmThermalFluid" , "Thrm"));
            names.push_back(std::make_pair("CmFunds"        , "Fund"));
            // loop thru 'names'
            std::pair<std::string, std::string> pair;  // [1]
            // [1] CAUTION: BOOST_FOREACH expects just one comma,
            // hence the use of a predeclared loop variable 'pair'
            BOOST_FOREACH( pair, names )
              {
                const std::string tag = boost::str(boost::format("I%d%sE")
                                                   % pair.first.length()
                                                   % pair.first );
                const std::string rep = ":" + pair.second;   // note colon
                if ( boost::ends_with(tid, tag) )
                  {
                    tid = xeona::trimLeadingDigits(tid);
                    boost::replace_last(tid, tag, rep);
                    break;
                  }
              }
            return tid;
          }
        else
          {
            return xeona::trimLeadingDigits(tid);
          }
      }

#else
# warning "g++ specific code omitted: function 'xeona::demangle' now ineffective"
    // this is only an issue of aesthetics, not substance --
    // whilst noting also that '#warning' is a GCC extension!

    else return tid;

#endif // def __GNUG__

  }

  // ------------------------------------------------------------
  //  EXPLICIT TEMPLATE INSTANTIATIONS
  // ------------------------------------------------------------

  //  See Cline (2006, section 35) for a simple explanation.  The
  //  topic is also covered more comprehensively by Lischner
  //  (2003 pp195-199).
  //
  //  Failure to instantiate template will typically result in a
  //  link-time error whenever a call is made.  The error usually
  //  says: "undefined reference to ...".  If no such call is
  //  ever made, then the program should compile and run without
  //  a murmur.
  //
  //    Cline, Marshall.  2006.  C++ FAQ lite.
  //      [www.parashift.com/c++-faq-lite/templates.html].
  //
  //    Lischner, Ray.  2003.  C++ in a nutshell : a language and
  //      library reference.  O'Reilly and Associates, Sebastopol,
  //      California, USA.  ISBN 0-596-00298-X.

  // FUNCTION 'xeona::vectorRepeat'

  template bool vectorRepeat<std::string>
  (const std::vector<std::string>&, std::vector<std::string>&);

  // FUNCTION 'xeona::tailCombine'

  // the following is just for unit test purposes
```

```
  typedef std::vector<int> vec0_type;
  template void tailCombine<vec0_type>(vec0_type&, const vec0_type&);

  typedef std::vector<shared_ptr<Record> > vec1_type;
  template void tailCombine<vec1_type>(vec1_type&, const vec1_type&);

  // FUNCTION 'xeona::mapCombine'

  // the following is just for unit test purposes
  typedef std::map<shared_ptr<Class>,
                   shared_ptr<Class> > map0_type;
  template void mapCombine<map0_type>(map0_type&, const map0_type&);

  typedef std::map<shared_ptr<Entity>,
                   shared_ptr<Entity> > map1_type;
  template void mapCombine<map1_type>(map1_type&, const map1_type&);

  typedef std::map<shared_ptr<Gateway>,
                   shared_ptr<DomainController> > map2_type;
  template void mapCombine<map2_type>(map2_type&, const map2_type&);

  typedef std::map<shared_ptr<Gateway>,
                   shared_ptr<Block> > map3_type;
  template void mapCombine<map3_type>(map3_type&, const map3_type&);

  typedef std::map<shared_ptr<Block>,
                   shared_ptr<DomainController> > map4_type;
  template void mapCombine<map4_type>(map4_type&, const map4_type&);

  // CAUTION: must place new code above the template instantiations

} // namespace xeona

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : util2.cc
//  file-create-date : Thu 06-Nov-2008 16:53 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions which offer general utilities 2 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/util2.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "util2.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting, max()
#include <iterator>           // STL additional iterators, std::distance()
#include <numeric>            // STL numerical algorithms
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <cmath>              // C-style maths, abs(), ceil(), floor(), sqrt()

#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/logic/tribool.hpp>      // three state boolean
#include <boost/logic/tribool_io.hpp>   // CAUTION: essential for std::boolalpha reporting

//  NAMESPACE DECLARATIONS

using boost::logic::tribool;            // three state boolean
using boost::logic::indeterminate;      // allows 'indeterminate' and 'indeterminate()'

//  CODE

namespace xeona
{

  // ----------------------------------------------------------
  //  FREE FUNCTION   : geometricProgression
  // ----------------------------------------------------------
  //  Description  : transform positive integer to geometric progression
  //  Role         : called by 'xeona::isTwoContained'
  //  Techniques   : arithmetic
  //  Status       : complete
  //
  //  Design notes
  //
  //      This routine uses conventional arithmetic, but it can
```

```
  //      also be done with bitwise operators (if I am not
  //      mistaken).  See commit r3831 for more information at to
  //      how.
  //
  //  Examples
  //
  //           0 -> empty vector
  //           1 -> { 1 }
  //          30 -> { 0  2  4  8 16 }
  //          32 -> { 0  0  0  0  0 32 }
  //
  // ------------------------------------------------------------

  std::vector<int>
  geometricProgression
  (const int aggregate)
  {
    // logger
    static logga::spLogger logger = logga::ptrLogStream();

    // geometric progression vector
    std::vector<int> reds;                  // vector of remainder values or zeros

    // integrity checks
    if ( aggregate < 0 )                    // non-negativity condition
      {
        logger->repx(logga::warn, "negative aggregate supplied", aggregate);
        logger->repx(logga::dbug, "abandoning function", "");
        return reds;
      }

    // establish an upper limit power-of-two
    int base = 1;
    while ( base < aggregate ) base *= 2;

    // fill the 'reds' vector
    int lid = base;
    int red = aggregate;                    // remainder value, reducing when needed
    while ( lid > 0 )
      {
        if ( red >= lid )
          {
            red = red % lid;                // modulo arithmetic operator, remainder
            reds.push_back(lid);            // .. after division
          }
        else
          {
#if 1 // 1 = omit "leading" (later "trailing") zeros, 0 = otherwise
            if ( ! reds.empty() ) reds.push_back(0);   // else omit "leading" zeros
#else
            reds.push_back(0);
#endif // 0
          }
        lid = lid / 2;                      // halving operation
      }

    // reverse order of elements
    std::reverse(reds.begin(), reds.end());  // refer <algorithm>

    // check the 'reds' vector sums to 'aggregate'
    const int sum = std::accumulate         // refer <numeric>
      (reds.begin(), reds.end(),            // range [first, last)
       0);                                  // initial value
    if ( aggregate != sum )
      {
        std::cerr << "** coding error 01 in source file " << __FILE__ << std::endl;
      }

    // additional reporting
    // YEEK 9 CODE (set by '--yeek')
    if ( xeona::yeek == 9 || xeona::yeek == 1 )
      {
        std::ostringstream put;
        put << std::boolalpha;
        put << "  yeek reporting (" << xeona::yeek << ")" << "\n"
            << "    aggregate  : "  << aggregate          << "\n"
            << "    base       : "  << base               << "\n"
            << "    vector     :";
        BOOST_FOREACH( int i, reds ) put << " " << i;
        put << "\n";
        put << "    sum        : "  << sum                << "\n";
```

```
      logger->putx(logga::yeek, put);
    } // if 'yeek 9'

  // return
  return reds;

} // function 'geometricProgression'

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::isTwoContained
// ----------------------------------------------------------
//  Description  : test if 'candidate' is a power-of-two sub-sequence of 'aggregate'
//  Role         : used for checking commitment strategy codes
//  On success   : returns 'true' if 'candidate' is contained in 'aggregate'
//  On fail      : returns 'indeterminate' for faulty input
//  Techniques   : 'std::set_difference'
//  Status       : complete
// ----------------------------------------------------------

tribool                                 // 'indeterminate' means faulty input
isTwoContained
(const int candidate,                   // greater than zero
 const int aggregate)                   // non-negative, can be less than 'candidate'
{
  // logger
  static logga::spLogger logger = logga::ptrLogStream();

  // basic range checks
  if ( candidate < 1 ) return indeterminate;    // 2
  if ( aggregate < 0 ) return indeterminate;    // 2

  // grab the geometric progressions
  std::vector<int> redCan = xeona::geometricProgression(candidate);
  std::vector<int> redAgg = xeona::geometricProgression(aggregate);

  // remove zeros and, in the process, produce sorted containers
  redCan.erase(std::remove(redCan.begin(), redCan.end(), 0), redCan.end());
  redAgg.erase(std::remove(redAgg.begin(), redAgg.end(), 0), redAgg.end());

  // vector for set-difference residuals
  std::vector<int> resids;

  // CAUTION: sorting: the 'set_difference' function template
  // from <algorithm> requires sorted ranges -- moreover here
  // the range is the entire container

  std::set_difference(redCan.begin(), redCan.end(),  // refer <algorithm>
                      redAgg.begin(), redAgg.end(),
                      std::back_inserter(resids));   // refer <iterator>, [1]

  // [1] results container: Josuttis (1999 p420) writes "the
  // caller must ensure that the destination range is big
  // enough or that insert iterators are used" -- the latter
  // approach is used here as also indicated on p272

  // additional reporting
  // YEEK 10 CODE (set by '--yeek')
  if ( xeona::yeek == 10 || xeona::yeek == 1 )
    {
      std::ostringstream put;
      put << std::boolalpha;
      put << "  yeek reporting (" << xeona::yeek << ")" << "\n"
          << "    function   : "  << __func__           << "\n"
          << "    candidate  : "  << candidate          << "\n"
          << "    aggregate  : "  << aggregate          << "\n"
          << "    return     : "  << resids.empty()     << "\n";
      put << "    dezeroed reduced candidate :";
      BOOST_FOREACH( int i, redCan ) put << " " << i;
      put << "\n";
      put << "    dezeroed reduced aggregate :";
      BOOST_FOREACH( int i, redAgg ) put << " " << i;
      put << "\n";
      put << "    residual   :";
      BOOST_FOREACH( int i, resids ) put << " " << i;
      put << "\n";
      logger->putx(logga::yeek, put);
    } // if 'yeek 10'

  // return
  if ( resids.empty() ) return true;        // 1
  else                  return false;       // 0
```

```
  } // function 'isTwoContained'

  // ------------------------------------------------------------
  //  FREE FUNCTION   : xeona::reducedVector
  // ------------------------------------------------------------

  std::string                                // no trailing newline
  reducedVector                              // wrapper-style call
  (const int         aggregate,
   const std::string separator)              // note default
  {
    const std::vector<int> reds = geometricProgression(aggregate);

    std::stringstream oss;
    BOOST_FOREACH( int i, reds )
      {
        oss << i << separator;
      }

    std::string buf = oss.str();
    const int len   = buf.size() - separator.size();
    buf = buf.substr(0, len);                // trim final separator

    return buf;
  }

} // namespace 'xeona'

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::coeffProduct <>
// ------------------------------------------------------------
//  Description  : calculate inner product while honoring constant term
//  Role         : obtain objective result without using the solver
//  Techniques   : 'std::inner_product'
//  Status       : complete
//
//  Equation
//
//      answer = a_0 + sum( a_i * x_i )   for i in { 1, .., n }
//
//          with 'x_0' ignored for calculation purposes
//          and the vectors 'a' and 'x' being of length n+1
//
//  Design notes
//
//      The STL 'std::inner_product' function is employed.  This
//      function is discussed by Lischner (2003 p628) and
//      Josuttis (1999 pp427-429).
//
//      The 'boost::numeric::ublas::inner_prod' function from the
//      'Boost.uBLAS' library could also have been used.
//
//      The N() is default constructed, so for type 'double' this
//      simply yields 0.0.
//
//      Although templated, this function will probably only be
//      instantiated with 'double.'
//
// ------------------------------------------------------------

namespace xeona
{
  template<typename N>                       // implicit instantiation supported
  N
  coeffProduct
  (const std::vector<N> a,                   // coefficients vector
   const std::vector<N> x)                   // variable values vector
  {
    // logger
    static logga::spLogger logger = logga::ptrLogStream();
    logger->repx(logga::adhc, "entering member function", "");

    // integrity checks
    const int alen = a.size();
    const int xlen = x.size();
    if ( alen != xlen                        // imbalance
         || alen < 1                         // empty
         || xlen < 1 )                       // empty
      {
        // warn and abandon
```

```
        std::ostringstream oss;
        oss << alen << " : " << xlen;
        logger->repx(logga::warn, "vector size problem, a : x", oss.str());
        logger->repx(logga::xtra, "returning default value", N());
        return N();
      }
    if ( x.front() != N() )
      {
        // warn and continue
        logger->repx(logga::warn, "vector x zeroth element not zero", x.front());
        logger->repx(logga::xtra, "continuing nevertheless", "");
      }

    // calculation
    const N inprod = std::inner_product(++a.begin(),   // see <numeric>, skip first
                                        a.end(),
                                       ++x.begin(),   // similarly skip first element
                                        a.front());  // grab the "shift" term
    return inprod;
  }

} // namespace 'xeona'

// ----------------------------------------------------------
//  EXPLICIT TEMPLATE INSTANTIATIONS
// ----------------------------------------------------------

template double xeona::coeffProduct(const std::vector<double>, const std::vector<double>);
template  float xeona::coeffProduct(const std::vector<float> , const std::vector<float>);
template    int xeona::coeffProduct(const std::vector<int>   , const std::vector<int>);

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : util3.cc
//  file-create-date : Thu 31-Dec-2009 00:18 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : free functions for floating point comparison / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/util3.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  Overview
//
//      This unit offers two robust floating point "approximately
//      equal" functions based on "integer" comparison.  One
//      takes an integral "closeness" metric and the other takes
//      a convenient enum-defined descriptive threshold.
//
//      The original code and method is from Bruce Dawson (cited
//      shortly).
//
//      Best practice floating point comparison uses the integral
//      distance for assessing closeness.  Consider:
//
//          *(int*)&f
//
//      This expression takes the address of 'float' value 'f',
//      treats it as an 'int' pointer, and then dereferences it
//      -- in doing so, 'f' effectively becomes an integer.
//
//      Two presumably close values can then be compared by
//      determining whether their integer difference falls within
//      some desired threshold -- this being an upper bound on
//      the number of distinct floating point representations
//      that can lie between them.  A threshold of zero is
//      acceptable and provides for an exact match as per
//      operator==.
//
//      This comparison is neither equivalent to absolute error
//      nor relative error.  But it does map quite closely to
//      relative error, albeit with a two-fold variability
//      depending on where specifically the floats lie in terms
//      of their exponent range.  Moreover it is totally robust
//      for near-zero and opposite sign comparisons.
//
//      The above description is based on 4-byte 'float' and
//      'int' types, but this code instead uses 8-byte 'double'
//      and 'long long' types.
//
//  Environmental requirements
//
//      First, sizeof(long long) == sizeof(double).
//
```

```
//      Second, conformance with ANSI C99, the C language
//      standard dating from 1999, is necessary.
//
//      Third, this unit was developed and tested on Intel
//      hardware.  Additional testing is recommended if this code
//      is ported to a different CPU architecture.
//
//  Original download
//
//      code archive  : ftp://ftp.cygnus-software.com/pub/comparecode.zip
//      download date : 29-Dec-2009
//      code source   : stand-alone file 'CompareAsInt.cpp'
//
//  Accompanying documentation
//
//      Dawson, Bruce.  2006.  Comparing floating point numbers.
//        Webpage (revision 35)
//        http://www.cygnus-software.com/papers/comparingfloats/
//        Comparing%20floating%20point%20numbers.htm[1]
//
//  Coding considerations
//
//      I tried moving from the C-style (T*) construct to
//      static_cast<T*> calls, but my compiler rejected this.
//
//      Integer literals of type 'long long' typically require
//      the (case-insensitive) postfix type specifier "LL".
//
//      This code does not use 'xeona' logging in order to
//      improve its potential reusability.  In addition, most, if
//      not all, of the local includes can be omitted.
//
//      Some more experimentation with the 'xeona::Precision'
//      integer literals could be useful.
//
//  References
//
//      Lischner, Ray.  2003.  C++ in a nutshell : a language and
//        library reference.  O'Reilly and Associates, Sebastopol,
//        California, USA.  ISBN 0-596-00298-X.

//  LOCAL AND SYSTEM INCLUDES

#include "util3.h"            // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <cmath>              // C-style maths, abs(), ceil(), floor(), sqrt()

#include <boost/static_assert.hpp>      //compile-time assertions

//  COMPILE-TIME ASSERTIONS

namespace xeona                             // see documentation about local namespaces
{
  BOOST_STATIC_ASSERT( sizeof(long long) == sizeof(double) );
}

// CAUTION: these assertions on failure (for my compiler at
// least) yield the rather cryptic error message:
//
//    "file.cc:00: error: invalid application of 'sizeof' to
//     incomplete type 'boost::STATIC_ASSERTION_FAILURE<false>'"
//
// This "sizeof" remark has nothing to do with the use of sizeof
// in the assert test!  In any case, the test does work.

//  PREPROCESSOR MACROS FOR TEST PURPOSES

#ifdef _XUTEST
# define XE_REPORTING 1
#else
# define XE_REPORTING 0 // 0 = silence, 1 = verbose
#endif
```

```
//  CODE

// for reference
// negative zero (hex)     = 0x8000000000000000LL (15 zeros)
// maximum long long (dec) = +9223372036854775807 (19 digits)

// ----------------------------------------------------------
//  FREE FUNCTION   : xeona::almostEqual (enum-based wrapper)
// ----------------------------------------------------------
//  Description  : compare two floats approximately using enum-based thresholds
//  Role         : general use, with support for predefined levels of precision
//  Techniques   : overloading, enum 'xeona::Precision'
//  Status       : complete
//
//  CAUTION: 'maxUlps' values
//
//      The hardcoded 'maxUlps' could benefit from more
//      development and testing -- although they are, in any
//      case, approximately correct.
//
// ----------------------------------------------------------

namespace xeona
{
  bool
  almostEqual
  (const double           A,
   const double           B,
   const xeona::Precision precision)         // see header
  {
    // determine max ulps value
    long long maxUlps;
    switch ( precision )
      {
      case xeona::exact: maxUlps =              0LL; break;
      case xeona::numic: maxUlps =     7600000000LL; break;
      case xeona::tight: maxUlps =   760000000000LL; break;
      case xeona::loose: maxUlps = 76000000000000LL; break;
      default:
        std::clog << "** coding error 01 in source file " << __FILE__ << std::endl;
        return false;
      }

    // workhorse call
    return xeona::almostEqual(A, B, maxUlps);
  }
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ::isInfinite (file-local)
// ----------------------------------------------------------
//  Description  : test for IEEE 754 positive or negative infinity
//  Role         : used by 'xeona::almostEqual'
//  Techniques   : inline, unnamed namespace
//  Status       : complete
// ----------------------------------------------------------

namespace
{
  inline
  bool
  isInfinite
  (double A)
  {
    return ( std::abs(A)                                    // refer <cmath>
             ==
             std::numeric_limits<double>::infinity() );     // refer <limits>

    // former code by Bruce Dawson retained for interest
    //
    // a 'float' type infinity has an exponent of 255 (shift left
    // 23 positions) and a zero mantissa -- there are also two
    // infinities: positive and negative
    //
    //     const int kInfAsInt = 0x7F800000;  // robbie: added the 'int'
    //     if ( (*(int*)&A & 0x7FFFFFFF) == kInfAsInt ) return true;
    //     else                                         return false;
  }
} // unnamed namespace
```

```
// ----------------------------------------------------------
//  FREE FUNCTION   : almostEqual (workhorse)
// ----------------------------------------------------------
//  Description  : compare two floats approximately using max-ulps value
//  Role         : general use, but with more control than  wrapper version
//  Techniques   : overloading, 'xeona::isInfinite', C-style casting, integer arithmetic
//  Status       : complete
//
//  Design notes
//
//      Preliminary checks
//
//          The 'maxUlps' cannot be negative but may be zero to
//          provide for an exact match
//
//          The original function from Bruce Dawson contained
//          three further checks based on: 'isInfinite', 'isNan',
//          and 'sign'.  Only one of these checks has been
//          retained here: the 'isInfinite' check.
//
//      Resolution issues
//
//          The 'double' type has a 53-bit mantissa and one ULP
//          (unit in the last place) implies a relative error of
//          between 1/4e20 and 1/8e20 (Dawson 2006).
//
//          The 'double' type std::numeric_limits<T>:: gives:
//
//              digits          :   53
//              digits10        :   15
//              max_exponent    : 1024
//              max_exponent10  :  308
//
//          See Lischner (2003) for further details on <limits>.
//
// ----------------------------------------------------------

namespace xeona
{
  bool
  almostEqual
  (const double     A,
   const double     B,
   const long long  maxUlps)                // maximum units of last place
    throw(std::domain_error)                // exception specification
  {
    // ---------------------------------
    //  input integrity (can throw)
    // ---------------------------------

    if ( maxUlps < 0 )
      {
        std::ostringstream oss;
        oss << "free function " << __func__ << " passed a negative maxUlps : " << maxUlps;
        throw std::domain_error(oss.str());  // see Lischner (2003 p658)
      }

    // ---------------------------------
    //  infinity check
    // ---------------------------------

    // if A or B are positive or negative infinity then only
    // return true if they are exactly equal to each other --
    // that is, if they are both infinities of the same sign --
    // this check is only needed if you will be generating
    // infinities and you don't want them 'close' to numbers near
    // std::numeric_limits<double>::max() (equivalently DBL_MAX)
    //
    // this test also prevents a sufficiently large 'maxUlps'
    // value from causing inf to -inf wrapping and hence a
    // spurious result (see the original code for more
    // information)

    if ( ::isInfinite(A) || ::isInfinite(B) )
      {
#if (XE_REPORTING == 1)
        std::cout << "infinities detected";
#endif
        return A == B;
      }
```

```
    // ---------------------------------
    //  core code
    // ---------------------------------

    // represent the two floats as integers (see information at
    // the head of file for the details)

    long long aInt = *(long long*)&A;
    long long bInt = *(long long*)&B;

    // make 'aInt' and 'bInt' lexicographically ordered as a
    // twos-complement integer -- the integer literal represents
    // negative zero in hex format

    if ( aInt < 0 ) aInt = 0x8000000000000000LL - aInt;
    if ( bInt < 0 ) bInt = 0x8000000000000000LL - bInt;

    // now compare 'aInt' and 'bInt' to find out how far apart
    // 'A' and 'B' are

    long long intDiff = aInt - bInt;         // get difference
    if ( intDiff < 0 ) intDiff = -intDiff;   // [1] absolute function for 'long long'

    // [1] CAUTION: integral functions 'abs' and 'labs' do not
    // support 'long long', hence the hard crafted code here

#if (XE_REPORTING == 1)
    std::cout <<  "maxUlps " << std::setw(18) << maxUlps
              << " intDiff " << std::setw(20) << intDiff
              << "  ";
#endif // XE_REPORTING

    // process and return result

    if ( intDiff <= maxUlps ) return true;
    else                      return false;
  }

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xemgen.cc
//  file-create-date : Mon 11-Aug-2008 14:46 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : class to generate well-formatted XEM models / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/xemgen.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "xemgen.h"           // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/foreach.hpp>            // BOOST_FOREACH iteration macro
#include <boost/format.hpp>             // printf style formatting

//  CODE

// ----------------------------------------------------------
//  CLASS           : XemGenerator
// ----------------------------------------------------------
//  Description  : generate well-formatted XEM model data
//  Role         : used by '--inbuilt' and various model testers
//  Techniques   : Boost.Format library
//  Status       : complete
// ----------------------------------------------------------

// STATIC DEFINITIONS

logga::spLogger XemGenerator::s_logger = logga::ptrLogStream();  // bind logger

// CREATORS

XemGenerator::XemGenerator
(const int tab) :                           // note default in header
  d_last(e_unknown),
  d_tab(tab),
  d_oss()
{
  s_logger->repx(logga::xtra, "constructor call, d_tab", d_tab);
}

XemGenerator::~XemGenerator()
{
```

```
  s_logger->repx(logga::adhc, "destructor call", "");
}

// RECORD LEVEL CALLS

void
XemGenerator::entity
(const std::string klass,
 const std::string identifier,
 const std::string preamble)                // defaults to "entity"
{
  d_oss << "\n";                            // always a blank line above
  d_oss << preamble << "." << identifier << "\n";
  d_last = e_record;
  in("class", klass);
  d_last = e_record;                        // override the value set by 'in'
}

void
XemGenerator::special
(const std::string special)
{
  d_oss << "\n";                            // always a blank line above
  d_oss << special << "\n";
  d_last = e_record;
}

void
XemGenerator::note()                        // usually followed by comment calls
{
  special("note");
}

// FIELD LEVEL CALLS

void
XemGenerator::in
(const std::string key,
 const std::string value)
{
  if ( d_last != e_input ) d_oss << "\n";
  d_last = e_input;
  field(key, ">", value);
}

void
XemGenerator::out
(const std::string key,
 const std::string value)
{
  if ( d_last != e_output ) d_oss << "\n";
  d_last = e_output;
  field(key, "<", value);
}

void
XemGenerator::inq
(const std::string key,
 const std::string quote)                         // quote as single string
{
  in(key, "\"" + quote + "\"");
}

void
XemGenerator::outq
(const std::string key,
 const std::string quote)
{
  out(key, "\"" + quote + "\"");
}

void
XemGenerator::inQ                           // multi-quote
(const std::string key,
 const std::string quote)                   // quote as timeseries string
{
  std::string buffer;
  std::vector<std::string> split;
  boost::split(split, quote, boost::is_any_of(" "), boost::token_compress_on);
  BOOST_FOREACH( std::string s, split )
    buffer += " \"" + s + "\"";                       // concatenate with quotes
```

```
  if ( ! buffer.empty() ) buffer = buffer.substr(1);   // chomp the first character
  in(key, buffer);                                     // secondary call
}

void
XemGenerator::outQ                              // multi-quote
(const std::string key,
 const std::string quote)                             // quote as timeseries string
{
  std::string buffer;
  std::vector<std::string> split;
  boost::split(split, quote, boost::is_any_of(" "), boost::token_compress_on);
  BOOST_FOREACH( std::string s, split )
    buffer += " \"" + s + "\"";                        // concatenate with quotes
  if ( ! buffer.empty() ) buffer = buffer.substr(1);   // chomp the first character
  out(key, buffer);                                    // secondary call
}

// PACKAGED CALLS

void
XemGenerator::horizon
(const int steps,
 const int interval)                            // note default in header
{
  entity("TimeHorizon" , xeona::timehorizon); // defined in 'common.cc'
  com("the TimeHorizon entity is REQUIRED and the");
  com("\"time-horizon\" identifier is MANDATORY");
  out("builtin-remark s", "");
  in("steps [-] i"     , boost::str(boost::format("%d") % steps));
  in("interval [s] i"  , boost::str(boost::format("%d") % interval));
  in("start-hour [-] i", "0");
  in("start-day [-] i" , "1");
  com("the start-hour begins midnight local time and ranges");
  com("[0,23] and the start-day begins 01-Jan and ranges [1,365]");
  inq("hemisphere s"   , "N");
  com("the hemisphere is {N,S} for north and south");
  com();
  com("the modeler should ensuring that timeseries data given");
  com("elsewhere aligns with the specification given here");
  com();
  com("header: b/builtins.h");
}

void
XemGenerator::overseer()
{
  entity("Overseer", xeona::overseer);
  com("the Overseer entity is REQUIRED and the \"overseer\"");
  com("identifier is MANDATORY");
  com();
  com("the overseer does little more that invoke the various");
  com("originating domains in nominated order at each new");
  com("interval");
  out("builtin-remark s", "");
  inq("captrans-algorithm s","simple");
  com("captrans-algorithm takes \"fixed\" | \"simple\" | \"hop-relit\"");
  com("but only \"simple\" is currently implemented (this call");
  com("contains experimental macro-controlled hop-relit code)");
  inq("ranked-orig-domains L", "domain-controller-1");
//inq("ranked-orig-domains L", "");
  com("the originating domain controllers must be given in");
  com("order of DESCENDING priority, any unannounced domains");
  com("will be discovered naturally during the various");
  com("traversals");
  out("total-financial [$] f"            , "0.0");
  out("total-greenhouse [kg] f"          , "0.0");
  out("total-nox [kg] f"                 , "0.0");
  out("total-depletion [J] f"            , "0.0");
  out("total-landuse [m^2] f"            , "0.0");
  com("the cost-type totals cover the entire horizon, with");
  com("first step truncation given by program.last-run.run-kind");
  // blanks(1); // used before the two 'com' calls were added
  out("variable-costs-financial [$] F"   , "0.0 ..");
  out("fixed-costs-financial [$] F"      , "0.0 ..");
  out("embedded-costs-financial [$] F"   , "0.0 ..");
  out("variable-costs-greenhouse [kg] F" , "0.0 ..");
  out("fixed-costs-greenhouse [kg] F"    , "0.0 ..");
  out("embedded-costs-greenhouse [kg] F" , "0.0 ..");
  out("variable-costs-nox [kg] F"        , "0.0 ..");
  out("fixed-costs-nox [kg] F"           , "0.0 ..");
```

```
  out("embedded-costs-nox [kg] F"         , "0.0 ..");
  out("variable-costs-depletion [J] F"    , "0.0 ..");
  out("fixed-costs-depletion [J] F"       , "0.0 ..");
  out("embedded-costs-depletion [J] F"    , "0.0 ..");
  out("variable-costs-landuse [m^2] F"    , "0.0 ..");
  out("fixed-costs-landuse [m^2] F"       , "0.0 ..");
  out("embedded-costs-landuse [m^2] F"    , "0.0 ..");
  com("header: b/overseer.h");
}

void
XemGenerator::emacs()
{
  std::string fill(xeona::modelFieldIndent, ' ');
  std::ostringstream otabs;
  otabs << "(" << boost::format("%02d") %  xeona::modelFieldIndent
        << " " << boost::format("%02d") %  d_tab
        << " " << boost::format("%02d") % (d_tab + 2)
        << ")";
  if ( d_last == e_record ) d_oss << "\n";
  d_last = e_comment;
  d_oss << fill << "useful emacs text editor settings"    << "\n"
        << fill << "local variables:"                     << "\n"
        << fill << "  mode: xem"                          << "\n"
        << fill << "  tab-stop-list: " << otabs.str()     << "\n"
        << fill << "  truncate-lines: t"                  << "\n"
        << fill << "end:"                                 << "\n";
}

// OTHER CALLS

void
XemGenerator::com()
{
  if ( d_last == e_comment ) d_oss << "\n";  // else do nothing
}

void
XemGenerator::com
(const std::string comment,
 const unsigned    pad)
{
  if ( d_last != e_comment ) d_oss << "\n";
  d_last = e_comment;
  d_oss << std::string(pad, ' ') << comment << "\n";
}

void
XemGenerator::hed
(const std::string header,
 const unsigned    pad)
{
  if ( d_last != e_comment ) d_oss << "\n";
  d_last = e_comment;
  d_oss << std::string(pad, ' ') << "header: " << header << "\n";
}

void
XemGenerator::ident                           // "Revision" to "    $Revision: 4800 $"
(const std::string key,
 const unsigned    pad)
{
  if ( d_last != e_ident ) d_oss << "\n";
  d_last = e_ident;
  d_oss << std::string(pad, ' ') << "$" << key << ": $" << "\n";
}

void
XemGenerator::meta                            // "role" to "xem-role:"
(const std::string key,
 const unsigned    pad)
{
  if ( d_last != e_meta ) d_oss << "\n";
  d_last = e_meta;
  d_oss << std::string(pad, ' ') << "xem-" << key << ":" << "\n";
}

void
XemGenerator::verbatim
(const std::string verbatim)
{
```

```
  if ( d_last != e_verbatim ) d_oss << "\n";
  d_last = e_verbatim;
  d_oss << verbatim << "\n";
}

void
XemGenerator::rule
(const std::string header)
{
  if ( d_last != e_rule ) d_oss << "\n";
  d_last = e_rule;
  const int space = 12;
  int chars = d_tab + space - header.length();
  d_oss << "  " << std::string(chars, '-') << " " << header << "\n";
}

void
XemGenerator::end()
{
  d_oss << "\n";
  d_oss << xeona::modelEndMarker << "\n";
  d_last = e_record;
}

void
XemGenerator::blanks
(const unsigned count)
{
  for ( unsigned int i = 0; i < count; ++i )
    {
      d_oss << "\n";
    }
}

// OUTPUT

void
XemGenerator::print
(std::ostream& os)
{
  os << d_oss.str()
     << std::flush;

  d_oss.str("");                            // purge local buffer
  d_last = e_unknown;                       // reset to unknown
}

std::string
XemGenerator::string()
{
  return d_oss.str();
}

// UTILITY FUNCTIONS

void
XemGenerator::field
(const std::string key,
 const std::string angle,
 const std::string value)
{
  std::ostringstream fmtss;                 // format string with embedded tab stop
  fmtss << "%|4t|%1% %|" << d_tab << "t|%2%";
  d_oss << boost::format(fmtss.str()) % key % angle;
  if ( ! value.empty() ) d_oss << " " << value;
  d_oss << "\n";
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : xeona_ptr.cc
//  file-create-date : Fri 31-Jul-2009 16:26 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : remappable counted pointer which mimics shared_ptr / implementation
//  file-status      : first-pass complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/c/xeona_ptr.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  File intentionally blank.

//  LOCAL AND SYSTEM INCLUDES

#include "xeona_ptr.h"        // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// CAUTION: intentionally blank, all code is in the header

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : glpkviz.cc
//  file-create-date : Thu 05-Jun-2008 14:03 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : HTML visualization of GLPK problem instances / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/d/glpkviz.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "glpkviz.h"          // companion header for this file (place first)

#if   (XE_GLPKVIZ_ALONE == 0)
#include "../d/siglp.h"       // semi-intelligent interface to GLPK MILP solver
#include "../c/files.h"       // free functions for regular files
#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)
#endif // XE_GLPKVIZ_ALONE

#include <algorithm>          // STL copying, searching, and sorting
#include <fstream>            // file-based io
#include <iomanip>            // setw() and family
#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <vector>             // STL sequence container

#include <cstdio>             // C-style io, remove(), EOF
#include <cstdlib>            // C-style exit(), getenv(), system(), NULL, EXIT_SUCCESS
#include <ctime>              // C-style time and date functions

#include <unistd.h>           // POSIX sleep(), usleep(), access(), chown()

#include <glpk.h>             // GNU GLPK mixed integer linear (MILP) solver

//  FILE-LOCAL TEST MACRO

//#define _DEBUG // table boundaries shown in light blue, system calls echoed

//  CODE

//  STATIC DEFINITIONS

std::string GlpkViz::s_helpExt   = "help";
std::string GlpkViz::s_htmlExt   = "html";
int         GlpkViz::s_callCount = 0;

#if   (XE_GLPKVIZ_ALONE == 1)               // stand-alone

  // CREATORS
```

```
GlpkViz::GlpkViz
(const std::string browserInvoke) :
  d_browser(browserInvoke),
  d_prob(NULL),
  d_html(),
  d_comment(),
  d_filename1(),
  d_filename2(),
  d_title(),
  d_round(true),
  d_timestamp()
{ }

GlpkViz::~GlpkViz()
{
}

// FUNCTION CALL OPERATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : operator()  (1 of 3)
// ----------------------------------------------------------
//  Description  : principal call for GLPK instances
//  Role         :
//  Signature    : 'glp_prob'
//  Techniques   :
//  Status       : complete
// ----------------------------------------------------------

void
GlpkViz::operator()
(glp_prob*         prob,                    // GLPK problem instance
 const std::string comment,
 const int         outputPrecision,
 const bool        glpkRounding)
{
  // CAUTION: it is not possible to make a logger call here

  // reset data members
  resetData();                              // utility call

  // grab problem
  d_prob = prob;
  if ( !d_prob ) return;

  // store comment
  d_comment = comment;

  // set output precision for floats
  d_html << std::setprecision(outputPrecision);

  // set close-to-zero rounding
  int prior = lpx_get_int_parm(d_prob, LPX_K_ROUND);
  lpx_set_int_parm(d_prob, LPX_K_ROUND, static_cast<int>(glpkRounding));

  // stamp the time
  d_timestamp = timestamp();                // utility call

  // main action
  prepareFilename();
  prepareTitle();
  createHtml();
  saveHtml(d_filename1);
  createHelp();
  saveHtml(d_filename2);
  displayHtml(d_filename1);

  // reset close-to-zero rounding
  lpx_set_int_parm(d_prob, LPX_K_ROUND, prior);

  return;
  }

#elif (XE_GLPKVIZ_ALONE == 0)               // 'xeona'

logga::spLogger GlpkViz::s_logger = logga::ptrLogStream();  // bind

// CREATORS

GlpkViz::GlpkViz
```

```
(const std::string browserInvoke) :            // thus a zero-argument ctor
  d_browser(browserInvoke),
  d_prob(NULL),
  d_html(),
  d_comment(),
  d_filename1(),
  d_filename2(),
  d_title(),
  d_round(true),
  d_timestamp()
{
  s_logger->repx(logga::dbug, "constructor call, browser string", d_browser);
}

GlpkViz::~GlpkViz()
{
  s_logger->repx(logga::dbug, "destructor call", "");
}

// ------------------------------------------------------
//  MEMBER FUNCTION : operator()  (2 of 3)
// ------------------------------------------------------
//  Description  : principal call for solver interface instances
//  Role         :
//  Signature    : SolverIf*
//  Techniques   :
//  Status       : complete
// ------------------------------------------------------

void
GlpkViz::operator()
(const svif::SolverIf* si,
 const std::string     comment,
 const int             outputPrecision,
 const bool            glpkRounding)
{
  s_logger->repx(logga::xtra, "entering function for raw pointer", "");

  // reset data members
  resetData();                              // utilty call

  // grab problem
  d_prob = si->d_prob;                      // this class has been granted friendship
  if ( !d_prob ) return;

  // store comment
  d_comment = comment;

  // set output precision for floats
  d_html << std::setprecision(outputPrecision);

  // set close-to-zero rounding
  int prior = lpx_get_int_parm(d_prob, LPX_K_ROUND);
  lpx_set_int_parm(d_prob, LPX_K_ROUND, static_cast<int>(glpkRounding));

  // stamp the time
  d_timestamp = timestamp();                // utility call

  // main action
  prepareFilename();
  prepareTitle();
  createHtml();
  saveHtml(d_filename1);
  createHelp();
  saveHtml(d_filename2);
  displayHtml(d_filename1);

  // reset close-to-zero rounding
  lpx_set_int_parm(d_prob, LPX_K_ROUND, prior);

  return;
}

// ------------------------------------------------------
//  MEMBER FUNCTION : operator()  (3 of 3)
// ------------------------------------------------------
//  Description  : wrapper to the SolverIf* principal
//  Role         :
//  Signature    : taking shared_ptr<svif::SolverIf>
//  Techniques   : 'shared_ptr::get'
//  Status       : complete
```

```
//
//  Design notes
//
//      Simple wrapper to the raw pointer version.  The
//      shared_ptr 'get' member function returns a raw pointer to
//      its controlled resource.
//
// ------------------------------------------------------

void
GlpkViz::operator()
(const shared_ptr<svif::SolverIf> si,        // solver interface object
 const std::string                comment,
 const int                        outputPrecision,
 const bool                       glpkRounding)
{
  s_logger->repx(logga::xtra, "entering wrapper for smart pointer", "");
  operator()(si.get(), comment, outputPrecision, glpkRounding);
}

#endif // XE_GLPKVIZ_ALONE

// UTILITY FUNCTIONS

void
GlpkViz::raw
(const std::string line)
{
  d_html << line << "\n";                    // simple insertion
}

void
GlpkViz::rem
(const std::string comment)
{
  d_html << "<!-- " << comment << " -->" << "\n";
}

void
GlpkViz::title
(const std::string pageTitle)
{
  std::string preamble;
#if 0 // 1 = use, otherwise 0
  preamble = "GLPK viz : ";
#endif // 0
  if ( pageTitle.empty() )
    d_html << "<title>" << preamble << "unnamed" << "</title>" << "\n";
  else
    d_html << "<title>" << preamble << pageTitle << "</title>" << "\n";
}

void
GlpkViz::title
(const char* info)                           // wrapper
{
  if ( info == NULL )                        // GLPK function returned char* NULL
    {
      title("");
    }
  else
    {
      std::string temp(info);                // convert to string
      title(temp);                           // make underlying call
    }
}

void
GlpkViz::h1
(const std::string header)
{
  if ( header.empty() )
    d_html << "<h1>" << "(unnamed)" << "</h1>" << "\n";
  else
    d_html << "<h1>" << header << "</h1>" << "\n";
}

void
GlpkViz::h1
(const char* header)                         // wrapper
{
```

```
  if ( header == NULL )                     // GLPK function returned char* NULL
    {
      h1("");
    }
  else
    {
      std::string temp(header);             // convert to string
      h1(temp);                             // make underlying call
    }
}

void
GlpkViz::rule()
{
  d_html << "<hr noshade size=1 align=\"left\" width=\"80%\">" << "\n";
}

void
GlpkViz::p
(const std::string   copy,
 const Embellishment e)
{
  std::string buf;
  switch ( e )
    {
    case none:     buf = "";                                  break;
    case helptext: buf = " class=\"helptext\"";               break;
    case endtext:  buf = " class=\"endtext\"";                break;
    default:
      std::clog << "** coding error 01 in source file " << __FILE__
                << ": given enum not supported on 'p' call: "
                << e
                << std::endl;
      break;
    }
  if ( copy.empty() )
    d_html << "<p" << buf << ">" << " " << "</p>" << "\n";
  else
    d_html << "\n" << "<p" << buf << ">" << "\n" << copy << "</p>" << "\n";
}

void
GlpkViz::p
(const char*         copy,                // wrapper
 const Embellishment e)
{
  if ( copy == NULL )                       // GLPK function returned char* NULL
    {
      p("", e);
    }
  else
    {
      std::string temp(copy);               // convert to string
      p(temp, e);                           // make underlying call
    }
}

void
GlpkViz::p
(const Embellishment e)                     // wrapper
{
  p("", e);
}

void
GlpkViz::line()
{
  d_html << "\n";                           // note leading newline
  d_html << "<tr>" << "\n";
}

void
GlpkViz::cell                               // blank cell wrapper
(const Embellishment e)                     // defaults to 'none'
{
  cell(" ", e);                        // note non-breaking space character entity
}

void
GlpkViz::cell                               // specialization to protect from char* NULL
(const char*         label,                 // char* wrapper
```

```
 const Embellishment e)                       // defaults to 'none'
{
  if ( label == NULL )                        // GLPK function returned char* NULL
    {
#if 1 // 1 = highlight unnamed row or col, 0 = do not highlight / user-modifiable
      cell(error);                            // only char* NULL and not empty string
#else
      cell();
#endif // 0
    }
  else
    {
      std::string temp(label);                // convert to string
      cell(temp, e);                          // make underlying call
    }
}

template <typename T>
void
GlpkViz::cell                                 // real cell
(const T             data,
 const Embellishment e)                       // defaults to 'none'
{
  std::string buf;
  switch ( e )
    {
    case none:    buf = "";                                    break;
    case narrow:  buf = " class=\"narrow\"";                   break;
    case label_c: buf = " class=\"label_c\"";                  break;
    case label_l: buf = " class=\"label_l\"";                  break;
    case kind_iv: buf = " class=\"kind-iv\"";                  break;
    case kind_bv: buf = " class=\"kind-bv\"";                  break;
    case meta:    buf = " class=\"meta\"";                     break;
    case meta_2:  buf = " class=\"meta_2\" colspan=\"2\"";     break;
    case zero:    buf = " class=\"zero\"";                     break;
    case result:  buf = " class=\"result\"";                   break;
    case goal:    buf = " class=\"goal\"";                     break;
    case optimal: buf = " class=\"optimal\"";                  break;
    case extra:   buf = " class=\"extra\"";                    break;
    case left:    buf = " class=\"left\"";                     break;
    case right:   buf = " class=\"right\"";                    break;
    case span_2:  buf = " class=\"stub\" colspan=\"2\"";       break;
    case error:   buf = " class=\"error\"";                    break;
    default:
      std::clog << "** coding error 02 in source file " << __FILE__
                << ": given enum not supported on 'cell' call: "
                << e
                << std::endl;
      break;
    }
  d_html << "<td" << buf << ">" << data << "\n";
}

void
GlpkViz::li
(const std::string item)
{
  if ( ! item.empty() )
    d_html << "<li>" << item << "</li>" << "\n";
}

void
GlpkViz::resetData()
{
  d_html.str("");                             // empty the non-const string-stream
  d_comment.clear();
  d_filename1.clear();
  d_filename2.clear();
  d_title.clear();
  d_timestamp.clear();
}

void
GlpkViz::prepareFilename()
{
  // inputs : GLPK problem name = "prob name", PID = 1234, call count = 2
  // ouput  : prob_name-001234-02.html

  const std::string htmlExt = "html";

  // CAUTION: the padding routines coded here require a positive
```

```
  // integer, otherwise the following can result: 000-22.  The
  // Boost.Format library provides a more robust solution (or a
  // more sophisticated routine could be hand coded).

  const int pidpad = 6;                       // NOTE: will NOT truncate bigger ints
  pid_t pid = getpid();                       // see <unistd.h>
  std::ostringstream ossPid;
  ossPid << std::setfill('0') << std::setw(pidpad) << static_cast<int>(pid);
  std::string paddedPid = ossPid.str();

  const int filepad = 2;
  std::ostringstream ossCnt;
  ossCnt << std::setfill('0') << std::setw(filepad) << ++s_callCount;
  std::string paddedCnt = ossCnt.str();

  std::string buf;
  const char* probname = glp_get_prob_name(d_prob);
  if ( probname != NULL )
    buf += probname;
  else
    buf += "unnamed";                         // not named default
  buf += "-" + paddedPid;
  buf += "-" + paddedCnt;

  // clean up the filename stub
  std::replace(buf.begin(), buf.end(), ' ', '_');      // space to underscore
  std::replace(buf.begin(), buf.end(), '.', '_');      // dot to underscore

  d_filename1  = buf;
  d_filename1 += "." + s_htmlExt;;

  d_filename2  = buf;
  d_filename2 += "." + s_helpExt;
  d_filename2 += "." + s_htmlExt;;
}

void
GlpkViz::prepareTitle()
{
  // output: 00 : prob name

  std::string probName;
  const char* probname = glp_get_prob_name(d_prob);
  if ( probname == NULL )
    probName = "(unnamed)";
  else
    probName = probname;

  // CAUTION: seen note elsewhere about this padding routine

  std::ostringstream oss;
  oss << std::setfill('0') << std::setw(2) << s_callCount;
  std::string paddedCallCount = oss.str();

  d_title += paddedCallCount;
  d_title += " : ";
  d_title += probName;
}

void
GlpkViz::createHelp()
{
  d_html.str("");                             // reset out-string-stream

  // -- start html ------------------------------------------

  raw("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">");

  raw("");
  std::ostringstream ossGlpkVer; ossGlpkVer << glp_version();    // returns char*
  std::ostringstream ossFile; ossFile << __FILE__;
  rem("GLPK viz help screen");
  rem("HTML creation " + d_timestamp);
  rem("C++ class 'GlpkViz' from file '" + ossFile.str() + "'");
  rem("GNU GLPK version " + ossGlpkVer.str());

  raw("");
  raw("<html>");
  raw("<head>");

  raw("");
```

```
  raw("<meta http-equiv=\"content-type\" content=\"text/html; charset=iso-8859-1\">");
  raw("<meta name=\"generator\" content=\"GLPK viz - 2008\">");
  raw("<meta name=\"description\" content=\"GLPK problem and solution visualization\">");

  raw("");
  title("GLPK viz : help");

  // STYLE

  raw("");
  raw("<style type=\"text/css\">");
  raw("/* HTML 4 tags */");
  raw("body          {font-family: sans-serif}");
  raw("h1            {color: #6c6753}");               // warm charcoal
  raw("h1            {padding-left: 2.0em}");
  raw("ul            {list-style-type: square}");
  raw("li            {padding-top: 1.0ex}");
  raw("tt            {font-weight: bold}");
  raw("tt            {color: #d54400}");               // dark orange
  raw("/* CSS1 anchor tag pseudo-classes */");
  raw("a:link        {color: #ffa500}");               // orange
  raw("a:visited     {color: #708090}");               // slate
  raw("/* CSS 1 classes */");
  raw("p.helptext    {font-size: normal; padding-top: 2.0ex}");
  raw("</style>");

  raw("");
  raw("</head>");

  // BODY

  raw("<body>");
  raw("");

  h1("GLPK viz : help");

  p("<b>Display format</b>", helptext);

  raw("");
  raw("<ul>");
  li("GLPK viz displays the problem in 'convenient' rather than 'standard' \
form &mdash; that is, without explicit auxiliary variables");
  raw("</ul>");

  p("<b>Main table</b>", helptext);

  raw("");
  raw("<ul>");
  li("all values are floats but decimal points are only displayed when needed");
  li("missing or suspect entries are indicated with an orange box");
  li("the <b>caller is responsible</b> for ensuring that the correct solvers \
have run and that the solution is current");
  li("results from the interior point solver <tt>glp_interior</tt> cannot be displayed");
  li("the objective 'shift' coefficient is shown in the \
<font class=\"nowrap\">(OBJ, RHS)</font> cell");
  li("any unboundedness comments derive from <tt>glp_get_unbnd_ray</tt>");
  raw("</ul>");

  p("<b>Abbreviations</b>", helptext);

  raw("");
  raw("<ul>");
  li("con = continuous structural variable");
  li("OBJ = objective function");
  li("BNDS = col bounds");
  li("recosts = LP reduced costs (dual values)");
  li("nan = not-a-number (zero divide zero) float");
  li("inf = infinity (divide zero) float");
  raw("</ul>");

  p("<b>Printing</b>", helptext);

  raw("");
  raw("<ul>");
  li("activate your browser 'print background' option to retain colors when printing");
  raw("</ul>");

  p("<b>About GLPK viz</b>", helptext);

  raw("");
  raw("<ul>");
```

```
  li("GLPK viz was written by Robbie Morrison");
  li("GLPK viz is licensed under <font class=\"nowrap\">GNU GPLv3</font>");
  raw("</ul>");

  raw("");
  p();
  rule();
  p();

  raw("");
  raw("</body>");
  raw("</html>");

  // -- end html ----------------------------------------
}

void
GlpkViz::createHtml()
{
  d_html.str("");                           // reset out-string-stream

  const int cols    = glp_get_num_cols(d_prob);   // col count
  const int rows    = glp_get_num_rows(d_prob);   // row count
  const int nzs     = glp_get_num_nz(d_prob);     // non-zero coefficients count
  const int unbnded = glp_get_unbnd_ray(d_prob);  // 0 or range [1, rows+cols]

  // -- start html ----------------------------------------

  // HEAD

  raw("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">");

  raw("");
  std::ostringstream ossGlpkVer; ossGlpkVer << glp_version();    // returns char*
  std::ostringstream ossFile; ossFile << __FILE__;
  rem("file generated automatically by interrogating a GLPK 'glp_prob' problem instance");
  rem("HTML creation " + d_timestamp);
  rem("C++ class 'GlpkViz' from file '" + ossFile.str() + "'");
  rem("GNU GLPK version " + ossGlpkVer.str());

  raw("");
  raw("<html>");
  raw("<head>");

  raw("");
  raw("<meta http-equiv=\"content-type\" content=\"text/html; charset=iso-8859-1\">");
  raw("<meta name=\"generator\" content=\"GLPK viz - 2008\">");
  raw("<meta name=\"description\" content=\"GLPK problem and solution visualization\">");

  raw("");
  title(d_title);                           // in format: [GLPK viz :] 00 : problem name

  // STYLE

  raw("");
  raw("<style type=\"text/css\">");
  raw("/* HTML 4 tags */");
  raw("body          {font-family: sans-serif}");
  raw("h1            {color: #6c6753}");              // warm charcoal
  raw("h1            {padding-left: 2.0em}");
  raw("table         {}");
  raw("tr            {}");
  raw("td            {text-align: center; white-space: nowrap}");
  raw("td            {padding-left: 1.0em; padding-right: 1.0em}");
  raw("td            {border-style: solid; border-width: thin; border-color: white}");
#ifdef _DEBUG
  raw("td            {border-color: cyan}");          // very useful for testing
#endif // 0
  raw("/* CSS1 anchor tag pseudo-classes */");
  raw("a:link        {color: #ffa500}");              // orange
  raw("a:visited     {color: #708090}");              // slate
  raw("/* CSS 1 classes */");
  raw("p.helptext    {font-size: smaller; padding-top: 2.0ex}");
  raw("p.endtext     {font-size: 65%; white-space: nowrap}");
  raw("font.nobr     {white-space: nowrap}");
  raw("table.first   {border-style: solid; border-width: thin; border-color: #cccccc}");
  raw("table.first   {padding: 10px}");
  raw("table.second  {font-size: smaller; padding-top: 0.0ex; padding-bottom: 2.0ex}");
  raw("td.narrow     {padding: 0.0em;}");
  raw("td.label_c    {font-size: smaller}");
  raw("td.label_l    {font-size: smaller; text-align: left}");
```

```
raw("td.meta       {background-color: #c8beb7}");    // warm gray
raw("td.meta_2     {background-color: #c8beb7}");    // warm gray
raw("td.zero       {background-color: #f2f2f2}");    // 5% gray
raw("td.kind-bv    {background-color: #ccff00}");    // lime green
raw("td.kind-iv    {background-color: #9ACD32}");    // mid-green
raw("td.result     {background-color: #fbec5d}");    // gold
raw("td.goal       {background-color: #6c6753; color: white; font-weight: bold}");
raw("td.optimal    {background-color: #ff9955}");    // orange
raw("td.extra      {background-color: #dde9af}");    // light green
raw("td.left       {text-align: left}");
raw("td.right      {text-align: right}");
raw("td.stub       {text-align: left; font-weight: bold; padding-top: 2.0em}");
raw("td.error      {border-color: #ffa500}");        // orange
raw("</style>");

raw("");
raw("</head>");

// BODY

raw("<body>");
raw("");
h1(d_title);

// TABLE ONE -- main data

// start table
raw("");
raw("<table class=\"first\" summary=\"GLPK problem visualization\">");

// optimization sense, col labels
line();
switch ( glp_get_obj_dir(d_prob) )         // optimization sense
  {
  case GLP_MAX: cell("maximize", goal); break;
  case GLP_MIN: cell("minimize", goal); break;
  default: cell(error); break;
  }
for (int col = 1; col <= cols; col++)
  {
    cell(glp_get_col_name(d_prob, col), label_c);    // col name
  }
cell();
cell();
cell(narrow);
cell();
cell();
cell();
std::ostringstream ossHelp;
ossHelp << "<a href=\"" << d_filename2 << "\">help</a>";
cell();
cell(ossHelp.str());

// col headers C1 to RHS
line();
std::ostringstream ossRxC1;
ossRxC1 << rows << " &times; " << cols;
cell(ossRxC1.str()); //  cell(glp_get_prob_name(d_prob), label_c);
for (int col = 1; col <= cols; col++)
  {
    std::ostringstream osscol;
    osscol << "C" << col;
    cell(osscol.str(), meta);
  }
cell("RHS", meta_2);
cell(narrow);
cell("LP solution", label_c);
cell("MIP solution", label_c);
cell("LP recosts", label_c);
cell();
cell();

// col kinds
line();
cell(nzs);
for (int col = 1; col <= cols; col++)
  {
    switch ( glp_get_col_kind(d_prob, col) )
      {
      case GLP_CV: cell("con");           break;
      case GLP_IV: cell("int", kind_iv);  break;
```

```
        case GLP_BV: cell("bin", kind_bv);  break;
        }
    }
  cell();
  cell("&mdash;");
  cell(narrow);
  cell();
  cell();
  cell("&larr; col kinds", label_c);
  cell();
  cell();

  // objective function
  line();
  cell("OBJ", meta);
  for (int col = 1; col <= cols; col++)
    {
      double objcoef = glp_get_obj_coef(d_prob, col);  // objective function coefficient
      if ( objcoef == 0.0 )
        cell(objcoef, error);
      else
        cell(objcoef);
    }
  cell();
  cell(glp_get_obj_coef(d_prob, 0));         // objective function 'shift' at index 0
  cell(narrow);
  cell(glp_get_obj_val(d_prob), goal);       // LP Z value
  cell(glp_mip_obj_val(d_prob), goal);       // MIP Z value
  cell("&larr; Z values", label_c);
  cell(glp_get_obj_name(d_prob), label_l);   // objective function name
  cell();

  // constraint matrix
  for (int row = 1; row <= rows; row++)
    {
      line();
      std::ostringstream ossrow;
      ossrow << "R" << row;
      cell(ossrow.str(), meta);
      for (int col = 1; col <= cols; col++)
        {
          double coeff = getConCoef(row, col);
          if ( coeff == 0.0 )
            {
              // select the treatment of cells containing zero / user-modifiable
              // cell()                         // do nothing
              // cell("&ndash");                // n-dash
              cell(zero);                       // CSS class 'td.zero'
            }
          else
            {
              cell(coeff);
            }
        }
      std::ostringstream ossrhs;
      std::string space = "  ";
      switch ( glp_get_row_type(d_prob, row) )
        {
        case GLP_LO:                          // glyph for >=
          cell("&ge;");
          cell(glp_get_row_lb(d_prob, row));
          break;
        case GLP_UP:                          // glyph for <=
          cell("&le;");
          cell(glp_get_row_ub(d_prob, row));
          break;
        case GLP_FX:                          // glyph for =
          cell("=");
          cell(glp_get_row_lb(d_prob, row)); // lower needed
          break;
        case GLP_FR:
          cell("free", error);
          cell();
          break;
        case GLP_DB:
          cell("[&nbps;]");
          ossrhs << glp_get_row_lb(d_prob, row)   // lower first
                 << space
                 << glp_get_row_ub(d_prob, row) ;
          cell(ossrhs.str());
          break;
```

```
        default:
          cell(error);
          break;
        }
      if ( row == unbnded )                  // uses 'glp_get_unbnd_ray'
        {
          cell("unboundedness", error);
        }
      else
        {
          cell(narrow);
        }
      cell(glp_get_row_prim(d_prob, row), result);     // LP solution
      cell(glp_mip_row_val(d_prob, row), result);      // MIP solution
      cell(glp_get_row_dual(d_prob, row), extra);      // LP reduced costs
      cell(glp_get_row_name(d_prob, row), label_l);    // row label
      cell(ossrow.str(), label_l);
    }

  // variable bounds
  line();
  cell("BNDS", meta);
  for (int col = 1; col <= cols; col++)
    {
      std::string space = "&amp;nbsp;&amp;nbsp;";
      std::ostringstream oss;
      switch ( glp_get_col_type(d_prob, col) )
        {
#if 1 // 1 is [] range style, 0 is < comparison style / user-modifiable
        case GLP_LO:
          oss << "["
              << glp_get_col_lb(d_prob, col)
              << ",&amp;nbsp;"
              << "&amp;#8734;"                  // infinity
              << "]";
          cell(oss.str());
          break;
        case GLP_UP:
          oss << "["
              << "&amp;minus;&amp;#8734;"            // minus infinity
              << ",&amp;nbsp;"
              << glp_get_col_ub(d_prob, col)
              << "]";
          cell(oss.str());
          break;
        case GLP_FX:
          oss << "["
              << glp_get_col_lb(d_prob, col)
              << ",&amp;nbsp;"
              << glp_get_col_ub(d_prob, col)
              << "]";
          cell(oss.str(), error);
          break;
        case GLP_FR:
          cell("free", error);
          break;
        case GLP_DB:
          oss << "["
              << glp_get_col_lb(d_prob, col) // lower first
              << ",&amp;nbsp;"
              << glp_get_col_ub(d_prob, col)
              << "]";
          cell(oss.str());
          break;
#else
        case GLP_LO:                         // glyph for >=
          oss << "&amp;ge;" << space << glp_get_col_lb(d_prob, col);
          cell(oss.str());
          break;
        case GLP_UP:                         // glyph for <=
          oss << "&amp;le;" << space << glp_get_col_ub(d_prob, col);
          cell(oss.str());
          break;
        case GLP_FX:                         // glyph for =
          oss << "=" << space << glp_get_col_lb(d_prob, col);
          cell(oss.str(), error);
          break;
        case GLP_FR:
          cell("free", error);
          break;
        case GLP_DB:
```

```
          oss << glp_get_col_lb(d_prob, col) << space  // lower first
              << "&le;" << space
              << glp_get_col_ub(d_prob, col);
          cell(oss.str());
          break;
#endif // 0
        default:
          cell(error);
          break;
        }
    }
  cell();
  cell("&mdash;");
  cell(narrow);
  cell();
  cell();
  cell();
  cell();
  cell();

  // often row
  line();
  cell();
  for (int col = 1; col <= cols; col++)
    {
      if ( col == (unbnded - rows) )          // uses 'glp_get_unbnd_ray'
        {
          cell("unboundness", error);
        }
      else
        {
          cell();
        }
    }
  cell();
  cell();
  cell(narrow);
  cell();
  cell();
  cell();
  cell();
  cell();

  // results / linear
  line();
  cell("LP solution", label_c);
  for (int col = 1; col <= cols; col++)
    {
      cell(glp_get_col_prim(d_prob, col), result);     // structural variable value
    }
  cell();
  cell();
  cell(narrow);
  switch ( glp_get_status(d_prob) )
    {
    case GLP_OPT:    cell("LP optimal", optimal);      break;    // optimal
    case GLP_FEAS:   cell("LP feasible");              break;    // feasible
    case GLP_INFEAS: cell("LP INFEAS", error);         break;    // solution infeasible
    case GLP_NOFEAS: cell("LP NOFEAS", error);         break;    // proven not feasible
    case GLP_UNBND:  cell("LP UNBND",  error);         break;    // unbounded
    case GLP_UNDEF:  cell("LP undefined",  error);     break;    // probably not run
    default: cell(glp_get_status(d_prob), error);      break;
    }
  cell();
  cell();
  cell();
  cell();

  // results / non-linear
  line();
  cell("MIP solution", label_c);
  for (int col = 1; col <= cols; col++)
    {
      cell(glp_mip_col_val(d_prob, col), result); // structural variable value
    }
  cell();
  cell();
  cell(narrow);
  cell();
  switch ( glp_mip_status(d_prob) )
    {
```

```
    case GLP_OPT:    cell("MIP optimal", optimal);     break;    // optimal
    case GLP_FEAS:   cell("MIP feasible");             break;    // feasible
    case GLP_NOFEAS: cell("MIP NOFEAS", error);        break;    // proven not feasible
    case GLP_UNDEF:  cell("MIP undefined",  error);    break;    // probably not run
    default: cell(glp_mip_status(d_prob), error);      break;
    }
  cell();
  cell();
  cell();

  // results / shadow
  line();
  cell("LP recosts", label_c);
  for (int col = 1; col <= cols; col++)
    {
      cell(glp_get_col_dual(d_prob, col), extra); // LP reduced cost
    }
  cell();
  cell();
  cell(narrow);
  cell();
  cell();
  cell();
  cell();
  cell();

  // col labels again
  line();
  cell();
  for (int col = 1; col <= cols; col++)
    {
      cell(glp_get_col_name(d_prob, col), label_c);    // col name
    }
  cell();
  cell();
  cell(narrow);
  cell();
  cell();
  cell();
  cell();
  cell();

  // col headers C1 to Ccols
  line();
  cell();
  for (int col = 1; col <= cols; col++)
    {
      std::ostringstream osscol;
      osscol << "C" << col;
      cell(osscol.str(), label_c);
    }
  cell();
  cell();
  cell(narrow);
  cell();
  cell();
  cell();
  cell();
  cell();

  // complete the table
  raw("");
  raw("</table>");

  // TABLE TWO -- further information

  raw("");
  raw("<table class=\"second\" summary=\"GLPK problem further information\">");

  line();
  cell("Additional GLPK details", span_2);   // only one cell is correct

  line();
  cell("simplex iterations", right);
  cell(lpx_get_int_parm(d_prob, LPX_K_ITCNT), left);    // lpx is correct

  line();
  cell("problem name", right);
  cell(glp_get_prob_name(d_prob), left);

  std::stringstream ossPtr; ossPtr << d_prob;
```

```
  line();
  cell("pointer address", right);
  cell(ossPtr.str(), left);

  std::string rounding;
  switch ( lpx_get_int_parm(d_prob, LPX_K_ROUND) )
    {
    case 0: rounding = "no";  break;
    case 1: rounding = "yes"; break;
    }
  line();
  cell("close&ndash;to&ndash;zero rounding (LPX_K_ROUND)", right);
  cell(rounding, left);

  std::ostringstream ossSize;
  ossSize << rows << " &times; " << cols;
  line();
  cell("constraint matrix (rows x cols)", right);
  cell(ossSize.str(), left);

  line();
  cell("non-zero coefficients", right);
  cell(nzs, left);

  double ratio1 =
    static_cast<double>(nzs)/static_cast<double>(rows);
  line();
  cell("non-zero coefficients/rows", right);
  cell(ratio1, left);

  double ratio2
    = static_cast<double>(nzs)/(static_cast<double>(rows) * static_cast<double>(cols));
  line();
  cell("non-zero coefficients/all coefficients", right);
  cell(ratio2, left);

  line();
  cell("GLPK version", right);
  cell(glp_version(), left);

  line();
  cell("HTML creation information", span_2);

  line();
  cell("associated comment", right);
  if ( !d_comment.empty() )
    cell(d_comment, left);
  else
    cell("(none)", left);

  line();
  cell("output (std::ostream) precision", right);
  cell(d_html.precision(), left);

  line();
  cell("saved filename (name + PID + 00)", right);
  cell(d_filename1, left);

  line();
  cell("HTML creation", right);
  cell(d_timestamp, left);

  raw("");
  raw("</table>");

  // FINAL RULE AND COMMENT

  raw("");
  rule();
  std::string final;
  final += "Machine generated HTML  &bull;  GLPK viz  &bull;  2008";
  p(final, endtext);

  raw("</body>");
  raw("</html>");

  // -- end html --------------------------------------------

  return;
}
```

```
std::string
GlpkViz::timestamp()
{
  // option A produces: 2008-Jun-02 15:28:46 UTC
  // NOTE: C rather that Boost time functions were chosen for portability
  const size_t MAX = 1024;                  // define buffer length
  const time_t now = time(0);               // seconds since 1970
  const tm* utc    = gmtime(&now);          // convert to tm struct in UTC time
  char timeStamp[MAX];

#if 1 // 0 = option B, 1 = option A / user-modifiable
  strftime(timeStamp, MAX, "%Y-%b-%d %H:%M:%S UTC", utc);
#else
  strftime(timeStamp, MAX, "%Y-%m-%d %H%:M:%S UTC", utc);   // ISO 8601
#endif // 0

  std::string buff(timeStamp);              // convert to std::string
  return buff;
}

double
GlpkViz::getConCoef
(int row,
 int col)
{
  // because there is no 'double glp_get_mat_val(glp_prob* lp, int i, int j)'

  int count = glp_get_mat_row(d_prob, row, NULL, NULL);     // non-zeros in given row

  const int offset = 1;                     // GLPK does not use index zero
  std::vector<int>    index(count + offset); // a 'reserve' call is not sufficient
  std::vector<double> value(count + offset);

  // Josuttis (1999 p155) writes: "whenever you need an array of
  // type T for any reason (such as for an existing C library)
  // you can use a std::vector<T> and pass it the address of the
  // first element".  Namely &vec[0].  The reason this works is
  // that std::vector elements, like C array elements, are
  // required to be held in contiguous memory.
  //
  //    Josuttis, Nicolai M.  1999.  The C++ Standard Library :
  //      a tutorial and reference.  Addison-Wesley, Boston, USA.
  //      ISBN 0-201-37926-0.

  glp_get_mat_row(d_prob, row, &index[0], &value[0]);

  // the following code is very STL-ish

  std::vector<int>::iterator        pos;
  std::vector<int>::difference_type idx;     // probably typedef'ed to int
  pos = std::find(index.begin(), index.end(), col);
  if ( pos != index.end() )                 // 'col' was found
    {
      idx = std::distance(index.begin(), pos);
      return value.at(idx);
    }
  else
    {
      return 0.0;
    }
}

void
GlpkViz::saveHtml
(const std::string filename)
{
  // write to file using truncate mode
  std::ofstream ofile(filename.c_str(), std::ios::out|std::ios::trunc);
  if ( !ofile )
    {
      std::clog << "** std::ofstream failed : " << filename << std::endl;
      return;
    }
  ofile << d_html.str();
  ofile.close();

#if   (XE_GLPKVIZ_ALONE == 0)               // not stand-alone
  xeona::readonly(filename);                // optional call, defined in unit 'c/util4'
#endif // XE_GLPKVIZ_ALONE

}
```

```
void
GlpkViz::displayHtml
(const std::string filename)
{
  // ----------------------------------------------------------
  //
  //  The following comments are based on the 'firefox'
  //  web-browser for Linux -- versions 2.0.0.13 and 3.6.6.
  //
  //  CAUTION: start 'firefox' first!
  //
  //  I tried 'firefox' options: (none), -safe-mode, and
  //  -no-remote on a dummy first call basis and nothing worked.
  //
  //  The 'firefox' warning dialog box says:
  //
  //     "Firefox is already running, but is not responding. To
  //     open a new window, you must first close the existing
  //     Firefox process, or restart your system."
  //
  //  The following made no difference also:
  //
  //      if ( s_callCount > 1 ) sleep(5);
  //
  //  I also experimented with the 'fork' function, but exiting
  //  cleanly is a problem.  And 'eval'ing under 'system':
  //
  //      eval "firefox sample-003442-01.html &"
  //
  // ----------------------------------------------------------

  // a one second delay means that the HTML pages open in order
  // (a distinct advantage) -- tested using a circa 2010 4-core
  // Intel Core i5 laptop

  const int delay = 1;                        // zero to disable
  if ( delay > 0 )
    {
      if ( s_callCount > 1 )
        {
          s_logger->repx(logga::adhc, "adding delay (to ensure HTML order)", delay);
          sleep(delay);
        }
    }

#if _XUTEST // 1 = disable browser call, 0 = enable
  return;
#endif // 0

  std::string browserCall;
  browserCall = d_browser + " \"" + filename + "\" &";      // filename set in soft quotes

#ifdef _DEBUG
  std::cout << "shell call : " << browserCall << std::endl;
#endif // 0

  int ret = system(browserCall.c_str());            // see <cstdlib>
  if ( ret != 0 )
    {
      std::clog << "** shell call failed : " << browserCall << std::endl;
    }
  return;
}

#undef _DEBUG // file local debug macro

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : siglp.cc
//  file-create-date : Tue 22-Apr-2008 14:37 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : semi-intelligent interface to GLPK MILP solver / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/d/siglp.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  List of depreciated GLPK APIs:
//
//      4.18: lpx_simplex
//
//      4.20: lpx_integer          / see: glp_iotopt
//
//      4.29: lpx_read_mps
//            lpx_read_freemps
//            lpx_write_mps
//            lpx_write_freemps
//            lpx_read_cpxlp
//            lpx_write_cpxlp
//
//      4.31: lpx_scale_prob
//            lpx_std_basis
//            lpx_adv_basis
//            lpx_cpx_basis
//
//      4.32: lpx_integer
//            lpx_intopt
//
//      4.33: lpx_exact
//            lpx_get_ray_info     / see: glp_get_unbnd_ray
//            lpx_interior
//            lpx_read_model
//
//      4.37: lpx_print_sol
//            lpx_print_ips        / see: glp_print_ipt
//            lpx_print_mip
//            lpx_print_prob       / see: glp_print_lp
//
//      4.38: glp_ipt_status       / not depreciated but two new return macros added
//
//      4.41: lpx_transform_row
//            lpx_transform_col
//            lpx_prim_ratio_test
//            lpx_dual_ratio_test
//
//      4.42: lpx_print_sens_bnds  / glp_print_ranges
//            glp_write_prob       / new call
//
//  AD-HOC NOTES
```

```
//
//  The 'xeona::readonly' calls are non-essential can be disabled
//  if need be.

//  LOCAL AND SYSTEM INCLUDES

#include "siglp.h"            // companion header for this file (place first)

#include "../c/files.h"       // free functions for regular files

#include "../a/logger.h"      // standard logging functionality (as required)
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <fstream>            // file-based io
#include <iomanip>            // setw() and family
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <ostream>            // output streams
#include <sstream>            // string-streams
#include <string>             // C++ strings
#include <cfloat>             // C-style DBL_MAX (1.79769e+308)
#include <cmath>              // C-style maths, ceil(), floor(), sqrt()
#include <cstdio>             // C-style io, remove(), EOF
#include <cstdlib>            // C-style exit(), getenv(), system(), NULL, EXIT_SUCCESS

#include <ctime>              // C-style time and date functions

#include <fcntl.h>            // C-style file control, manipulate file descriptor flags
#include <fenv.h>             // C-style C99 and TR1 floating point environment
#include <sys/stat.h>         // C-style POSIX file characteristics, stat()
#include <unistd.h>           // POSIX sleep(), usleep(), access(), chown()

#include <boost/algorithm/string.hpp>             // string recasing, trimming, splitting
#include <boost/algorithm/string_regex.hpp>       // additional regex support
#include <boost/foreach.hpp>                      // BOOST_FOREACH iteration macro

//  INTERNAL MACRO SETTING

#define STD_OSTREAM std::clog           // this aligns with current logging practice [1]

// [1] this should really be a static variable and not a pre-processor macro

#if !defined(XE_BNDS_DEFAULT)           // gives possibilty of external override
# define XE_BNDS_DEFAULT 1              // 0 = free, 1 = non-negative / user-modifiable
#endif

#if   (XE_BNDS_DEFAULT == 0)
# warning "XE_BNDS_DEFAULT macro set to free (which may be correct)"
#elif (XE_BNDS_DEFAULT == 1)
#else
# error "XE_BNDS_DEFAULT macro not set to a supported value {0,1}"
#endif

// MACRO-SET CONSTANT

namespace svif
{
  //  DEFINITIONS: debug flag
  //  controlled by compile-time _XDEBUG macro

#ifdef _XDEBUG                                // xeona macro
  const bool DBUG = true;
#else
  const bool DBUG = false;
#endif

} // namespace svif

//  CODE

namespace
{
  // ------------------------------------------------------------
  //  LOCAL CONSTANT  : ::len
  //  LOCAL CONSTANT  : ::PRO
  //  LOCAL CONSTANT  : ::OBJ
  //  LOCAL CONSTANT  : ::CON
  //  LOCAL CONSTANT  : ::VAR
  //  LOCAL CONSTANT  : ::VAZ
  //  LOCAL CONSTANT  : ::VAB
  //  LOCAL CONSTANT  : ::SEP
```

```
  // ------------------------------------------------------
  //  Description  : string constants for use in standardized labeling
  //  Role         : support
  //  Note         : modifiable by users, set to empty string to disable
  //
  //  CAUTION: all prefix strings must be of length 'len'
  //
  // ------------------------------------------------------

  const int         len = 3;          // CAUTION: fixed string length required
  const std::string PRO = "pro";      // problem prefix / user-modifiable
  const std::string OBJ = "obj";      // objective function prefix / user-modifiable
  const std::string CON = "con";      // constraint equation prefix / user-modifiable
  const std::string VAR = "var";      // real-valued variable prefix / user-modifiable
  const std::string VAZ = "vaz";      // integer-valued variable prefix / user-modifiable
  const std::string VAB = "vab";      // 0-1-valued prefix (binary) / user-modifiable
  const std::string SEP = ".";        // separating string (say . or -) / user-modifiable

  // ------------------------------------------------------
  //  FREE FUNCTION   : ::trimPrefixPlus
  // ------------------------------------------------------
  //  Description  : for example, left trim "var01." from "var01.more.details"
  //  Role         : local helper function
  //  Techniques   : Boost.String_alg library and 'regex'
  //  Status       : complete
  // ------------------------------------------------------

  bool                                  // 'true' indicates change made
  trimPrefixPlus
  (std::string& tag)                    // non-const pass-by-ref
  {
    // logging support
    static logga::spLogger logger = logga::ptrLogStream();  // main function logger

    // preamble
    const std::string original = tag;       // keep for later comparison

    // hardcoded prefix list
    const std::string prefixes[] = { ::PRO, ::OBJ, ::CON, ::VAR, ::VAZ, ::VAB };
    std::string separator       = ::SEP;

    // modify the separator if need be: a plain dot has special
    // meaning and needs to be escaped -- and the first escape
    // slash needs to be escaped again for C++ (read the manual!)
    if ( separator == "." ) separator = "\\.";

    // active code
    BOOST_FOREACH( std::string prefix, prefixes )
      {
        const std::string str = "^" + prefix + "[[:digit:]]*" + separator;
        const boost::regex pattern(str);
        boost::algorithm::erase_regex(tag, pattern);
        if ( original != tag )
          {
#if 1 // 0 = no reporting, 1 = use reporting
            logger->repx(logga::adhc, "current pattern", pattern);  // 'regex's stream
#endif // 0
            return true;
          }
      }

    // no match code
#if 0 // 0 = no reporting, 1 = use reporting
    logger->repx(logga::adhc, "no trim applied", original);
#endif // 0

    return false;

  } // function '::trimPrefixPlus'

  // ------------------------------------------------------
  //  FREE FUNCTION   : ::termHook
  // ------------------------------------------------------
  //  Description  : insert GLPK message into our nominated std::ostream
  //  Role         : used in combination with 'glp_term_hook'
  //  Techniques   : C hook function, void* (really!)
  //  Status       : complete
  // ------------------------------------------------------

  int
  termHook
```

```
(void*       info,                          // pointer to ostream as it happens
 const char* s)                             // information to be captured as C-string
{
  std::ostream* os = static_cast<std::ostream*>(info);    // cast from void*
  *os << s;                                 // need to dereference os
  return 1;                                 // non-zero to suppress normal GLPK output
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ::seq
// ----------------------------------------------------------
//  Description  : creates a sequence of integers for use in GLPK APIs
//  Role         : use in 'resetProblem'
//  Techniques   : std::vector
//  Status       : complete
//
//  Design notes
//
//      The call 'seq(2, 4)' returns a vector containing:
//
//          { 2, 3, 4 }
//
//      STL 'std::vector' vectors can be used as C arrays thus.
//      If a function takes a C array pointer, simply pass in
//      the address of the first element of the vector instead:
//
//          &vec[0]
//
//      See also
//
//          Lischner (2003 pp338-339) shows how to code the
//          following using the 'std::generate' algorithm.
//
// ----------------------------------------------------------

std::vector<int>                            // can be an empty vector
seq
(const int lower,                           // lower element
 const int upper)                           // upper element
{
  static logga::spLogger logger = logga::ptrLogStream();  // bind logger

  // passive integrity check
  if ( lower >= upper )
    {
      std::ostringstream oss;
      oss << lower << " : " << upper;
      logger->repx(logga::warn, "mismatch, lower : upper", oss.str());
    }

  // active code
  std::vector<int> vec;
  for ( int i = lower;                      // start value
        i <= upper;                         // end value
        ++i )                               // simple progression
    {
      vec.push_back(i);
    }
  return vec;
}

// ----------------------------------------------------------
//  FREE FUNCTION   : ::isIntegerValued
// ----------------------------------------------------------
//  Description  : test whether a variable bound is integer-valued
//  Role         : integrity check
//  Techniques   : 'floor'
//  Status       : complete
//
//  Design notes
//
//      Uses the 'floor' test from <cmath>.  This gave the same
//      apparent behavior as the following double cast test:
//
//          x == double(int(x))  // but C-style casts not recommended
//
//     Note the following result:
//
//         double eps = std::numeric_limits<double>::epsilon();
//         isIntegerValued(3.0 + eps) returns 'true'
//         isIntegerValued(3.0 + 1.1 * eps) returns 'false'
```

```
  //
  //    On IA32 systems (at least on mine), eps is 2.22045e-16.
  //
  // ----------------------------------------------------------

  bool                                      // output, 'true' means integer-valued
  isIntegerValued
  (double x)                                // input as double
  {
    return ( x == std::floor(x) ) ? true : false;
  }

} // unnamed namespace

namespace svif
{
  // ----------------------------------------------------------
  //  FREE FUNCTION   : svif::refDumpStream
  // ----------------------------------------------------------
  //  Description  : used to dump common ouput
  //  Role         : used by 'outputSuspend' and 'outputResume'
  //  Techniques   : free function with static data
  //  Status       : complete
  //
  //  Design notes
  //
  //      Curiously, one can write to a closed stream as a way of
  //      disposing of data.  For instance, the following is
  //      seems fully acceptable:
  //
  //          std::ofstream null("");
  //          null.is_open();              // returns 'false'
  //          null << "hi" << std::endl;
  //
  //      The use of static data means that the same stream is
  //      always returned.
  //
  // ----------------------------------------------------------

  std::ofstream&
  refDumpStream()
  {
#if 1 // 1 = dump to waste, 0 = redirect to file / user-modifiable
    static std::ofstream s_stream("");
#else
    const std::string filename = "sos-dump.log";
    static std::ofstream s_stream(filename.c_str(), std::ios::out|std::ios::trunc);
    if ( !s_stream )
      {
        std::cout << std::flush;
        std::clog << "** file open failed : " << s_stream << std::endl;
      }
#endif // 0
    return s_stream;
  }

  //  STATIC DEFINITIONS

  std::ostream*   SolverIf::s_os            = &STD_OSTREAM;
  bool            SolverIf::s_noise         = true;    // mostly passive
  int             SolverIf::s_instanceCount = 0;       // initially zero
  logga::spLogger SolverIf::s_logger        = logga::ptrLogStream();  // bind

  // CREATORS

  // ----------------------------------------------------------
  //  CONSTRUCTOR     : SolverIf
  // ----------------------------------------------------------
  //  Description  : construct solver interface object for client code
  //  Role         : creator
  //  Techniques   : calls 'utilCreateGlpkProb' to make GLPK problem instance
  //  Status       : complete
  //
  //  Design notes
  //
  //      Sparse matrix vectors 'd_ia', 'd_ja', and 'd_ar' are
  //      initialized here.  The index-zero (first) element is
  //      not used in GLPK (excepting obj_0 which is the
  //      objective 'shift'), hence the rather unusual
  //      initialization calls.  See also comments in the code.
  //
```

```
  // ----------------------------------------------------------

  SolverIf::SolverIf
  (std::string    tag,                      // problem name tag
   ReportingLevel noise) :

    // set fixed member variables
    d_probTag(tag),

    // set defaults for status variables
    d_employScaling(false),                 // scale problem
    d_employAdvBasis(false),                // use advanced initial LP basis
    d_simplexPresol(false),                 // simplex presolver
    d_integerPresol(false),                 // MIP presolver

    d_probStatus(status_not_specified),     // determined in this code
    d_objSense(sense_not_specified),        // must be set in client code
    d_probKlass(prob_not_specified),        // determined in this code
    d_solverType(solver_other),             // acceptable default behavior is provided
    d_solnStatus(soln_undefined),           // usefulness of solution
    d_solverRet(0),                         // if set, GLPK three-digit int
    d_solnRet(0),                           // if set, GLPK three-digit int

    // set problem administration information
    d_noise(noise),                         // currently no provision to reset
    d_solverCalls(0),
    d_alertCount(0),                        // interface thru 'incrementAlertCount'

    // prepare sparse matrix vectors -- pad the start of matrix
    // with zeros as required, zeros are fine because they are
    // illegal as indices and ineffective as coefficient values
    d_ia(s_glpkMargin, 0),
    d_ja(s_glpkMargin, 0),
    d_ar(s_glpkMargin, 0.0),

    // set GLPK problem instance pointer to null
    d_prob(NULL),
    d_parmSimplex(),                        // struct holding parameters
    d_parmInterior(),
    d_parmInteger()

  {
#if 0 // 1 = reserve space in constructor, 0 = omit

    const int SIZE = 1000;
    d_ia.reserve(SIZE);
    d_ja.reserve(SIZE);
    d_ar.reserve(SIZE);
    s_logger->repx(logga::info, "set reserve space in 3 arrays", SIZE);

#endif // 0

    // set reporting level as required (whilst noting that
    // logging verbosity is CONTROLLED by the app)
    if ( d_noise == svif::not_specified )    // meaning that the constructor defaulted
      {
        d_noise                   = svif::low;   // user-modifiable
        if ( svif::DBUG ) d_noise  = svif::high;  // user-modifiable
      }

    // initial reporting, including debug status
    s_logger->repx(logga::dbug, "constructor call", "");

    // default initialize the GLPK parameter structs
    glp_init_smcp (&d_parmSimplex);
    glp_init_iptcp(&d_parmInterior);
    glp_init_iocp (&d_parmInteger);

    // bind GLOBAL GLPK output to our nominated std::ostream 's_os'
    glp_term_hook(::termHook, s_os);        // void* type

    // create GLPK problem
    utilCreateGlpkProb(d_probTag);

    // overwrite GLPK defaults with the values hardcoded in this
    // file
    //
    // CAUTION: the outcome can depend on the value of
    // 'svif::DBUG' so read the code for the following call
    //
    utilSetGlpkDefaults();
```

```
    // defensive programming (later on, even addressing 'd_prob'
    // can segfault)
    if ( d_prob == NULL )
      {
        s_logger->repx(logga::warn, "NULL problem unexpected, d_prob", d_prob);
      }

    // completion reporting
    s_logger->repx(logga::adhc, "leaving member function", "");

  } // constuctor 'SolverIf'

  // -------------------------------------------------------
  //  DESTRUCTOR      : ~SolverIf (non-virtual)
  // -------------------------------------------------------
  //  Description  : destructor
  //  Role         : (un)creator
  //  Techniques   : calls 'utilDeleteGlpkProb' to clean up GLPK problem instance
  //  Status       : complete
  //
  //  CAUTION: free GLPK environment
  //
  //      This destructor assumes that there will be no
  //      non-SolverIf (rogue) GLPK problem instances in
  //      existence.  Otherwise disable the 'freeGlpkEnvironment'
  //      call and manage the issue elsewhere.
  //
  // -------------------------------------------------------

  SolverIf::~SolverIf()
  {
    // intial reporting
    s_logger->repx(logga::adhc, "destructor call", "");

    // flush local reporting stream
    *s_os << std::flush;                    // CAUTION: different from the logging stream

    // delete GLPK problem
    utilDeleteGlpkProb();                   // decrements GLPK problem instance count

#if 1 // 1 = free GLPK environment, 0 = disable / user-modifiable

    // free GLPK environment by calling a public member function
    // which then calls 'glp_free_env'
    if ( s_instanceCount == 0 )
      {
        s_logger->repx(logga::dbug, "about to call freeGlpkEnvironment", "");
        freeGlpkEnvironment();
      }

#endif // 0

    // report appropriately
    if ( d_alertCount == 0 )
      {
        s_logger->repx(logga::dbug, "destructor call, alert count", d_alertCount);
      }
    else
      {
        s_logger->repx(logga::info, "destructor call, alert count", d_alertCount);
      }

  } // destructor 'SolverIf'

  // APPLICATION CLEAN-UP

  // -------------------------------------------------------
  //  MEMBER FUNCTION : freeGlpkEnvironment (static)
  // -------------------------------------------------------
  //  Description  : frees the GLPK library environment
  //  Role         : application clean-up
  //  Techniques   : 'glp_free_env'
  //  Status       : complete
  //
  //  Design notes
  //
  //      A memory leak of 140 bytes currently results on app
  //      exit if not called.  'valgrind' notes:
  //
  //          "_glp_lib_init_env (glplib04.c:65)"
```

```
//          "still reachable: 140 bytes in 1 blocks"
//
//  CAUTION: call placement
//
//      Called when the 'SolverIf' destructor is invoked on the
//      last instance.
//
//      But if employed, use only near the end of the 'main'
//      function because any existing GLPK problem instances
//      will be unceremoniously "invalidated".
//
// ------------------------------------------------------------

bool                                    // 'false' if problems encountered
SolverIf::freeGlpkEnvironment()         // calls 'glp_free_env', warns if instances
{
  bool ret = true;                      // return value
  s_logger->repx(logga::info, "about to call GLPK glp_free_env", "");
  if ( s_instanceCount != 0 )
    {
      s_logger->repx(logga::warn, "will clobber problem instances", s_instanceCount);
      ret = false;
    }
  glp_free_env();                       // GLPK API, void return
  return ret;
}

// REPORTING CONTROL : class-wide calls controlling native GLPK
// output -- as opposed to xeona-style logging

// ------------------------------------------------------------
//  MEMBER FUNCTION : stdoutRedirect (static)
// ------------------------------------------------------------
//  Description  : permanently redirect stdout
//  Role         : application-wide reporting control
//  Techniques   : C-style UNIX code
//  Status       : complete
//
//  Design notes
//
//      This call is a hack!  Ideally the block on/off
//      functions should be implemented using 's_os.'
//
//      The code is based on Robbins and Robbins (2003 p131).
//      Who, in turn, highly recommend Stevens (1992) for UNIX
//      io issues.
//
//      All the system calls employed are POSIX-conforming.
//
//  CAUTION: global and permanent
//
//      The final splash screen will be affected.
//
//  References
//
//      Robbins, Kay A and Steven Robbins.  2003 UNIX systems
//        programming : communication, concurrency, and threads
//        -- Second edition.  Prentice Hall PTR, Upper Saddle
//        River, New Jersey, USA.  ISBN 0-13-042411-0.
//
//      Stevens, Richard W.  1992.  Advanced programming in the
//        UNIX environment.  Addison-Wesley, Reading,
//        Massachusetts, USA.
//
// ------------------------------------------------------------

void
SolverIf::stdoutRedirect
(const std::string logname)
{
  //  CAUTION: the POSIX macros S_IRGRP and S_IROTH are not
  //  defined in Windows as there is no way to deal with
  //  permissions for group and others.

  s_logger->repx(logga::dbug, "entering member function", "");

  if ( logname.empty() )
    {
      // simply close stdout
      if ( close(STDOUT_FILENO) == -1 )    // macro defined in <unistd.h>
        {
```

```
            perror("perror: failed to close stdout / system message");
            return;
          }
      }
    else // logname is not empty
      {
        // open file
        int fd;                               // POSIX UNIX-style file descriptor
        fd = open(logname.c_str(),            // POSIX UNIX-style io call
                  O_WRONLY | O_CREAT | O_APPEND,  // file create flags
                  S_IRUSR | S_IWUSR);             // file create mode
        if ( fd == -1 )                       // on fail, 'open' also sets 'errno'
          {
            std::ostringstream oss;
            oss << "perror: "
                << "failed to open file: "
                << logname
                << "  / system message";
            perror(oss.str().c_str());        // POSIX extension CX from <cstdio>
            return;
          }
        else
          {
            std::ostringstream oss;
            oss << "opened file descriptor " << fd;
            s_logger->repx(logga::dbug, oss.str(), logname);
          }

        // close stdout (optional, 'dup2' does this anyway, but also omits error messages)
        if ( close(STDOUT_FILENO) == -1 )     // macro defined in <unistd.h>
          {
            perror("perror: failed to close stdout / system message");
            return;
          }

        // remap stdout to file
        if ( dup2(fd, STDOUT_FILENO) == -1 ) // POSIX call
          {
            std::ostringstream oss;
            oss << "perror: "
                << "failed to redirect stdout to: "
                << logname
                << "  / system message";
            perror(oss.str().c_str());        // POSIX extension CX from <cstdio>
            return;
          }
        else
          {
            std::ostringstream oss;
            oss << "redirecting stdout " << STDOUT_FILENO << " to";
            s_logger->repx(logga::info, oss.str(), logname);
          }

      } // logname not empty

    return;

  } // function 'SolverIf::stdoutRedirect'

  // -------------------------------------------------------
  //  MEMBER FUNCTION : outputSuspend (static)
  //  MEMBER FUNCTION : outputResume  (static)
  // -------------------------------------------------------
  //  Description  : toggle GLPK output and native reporting
  //  Role         : class-wide reporting control
  //  Techniques   : 'glp_term_out', 's_noise' (as opposed to 'd_noise')
  //  Status       : incomplete
  //
  //  Design notes
  //
  //      These static functions toggle:
  //
  //          * the GLPK output which results from GLPK API calls
  //          * native reporting results from local calls
  //
  //      but do not affect:
  //
  //          * normal logging
  //
  //      The actions are global and affect all 'SolverIf'
  //      instances.
```

```
//
//      However, normal logging is controlled by the app in
//      association with the 'logger' unit.
//
//  CAUTION: GLPK environment needed
//
//      The call to 'glp_term_out' requires an existing GLPK
//      library environment but not necessarily any live GLPK
//      problem instances.  Hence the protection below and the
//      delayed call in 'utilCreateGlpkProb' when needed.
//
// ----------------------------------------------------------

void
SolverIf::outputSuspend()
{
  // CAUTION: 's_os' is a stream pointer and not a stream reference
  s_os = &svif::refDumpStream();           // free function with static data
  s_logger->repx(logga::info, "common out bound to closed stream", s_os);
  s_noise = false;

  // the following should only be seen by the dump stream
  *s_os << "** new redirect order" << std::endl;
}

void
SolverIf::outputResume()
{
  // CAUTION: 's_os' is a stream pointer and not a stream reference
  s_os = &STD_OSTREAM;
  s_logger->repx(logga::info, "common out bound to STD_OSTREAM", s_os);
  s_noise = true;
}

// SOLVER PREFERENCES : instance-based

// ----------------------------------------------------------
//  MEMBER FUNCTION : initSetPrefLPSolver
// ----------------------------------------------------------
//  Description  : choose between simplex and interior point solver
//  Role         : initialization method, optional
//  Techniques   : internal enum
//  Status       : complete
// ----------------------------------------------------------

void
SolverIf::initSetPrefLPSolver
(SolverType solverType)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  d_solverType = solverType;
  s_logger->repx(logga::dbug, "preferred solver set, solver type", d_solverType);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : initEmployScaling
// ----------------------------------------------------------
//  Description  : elect to use problem scaling
//  Role         : initialization method, optional
//  Techniques   : 'd_employScaling'
//  Status       : complete
// ----------------------------------------------------------

void
SolverIf::initEmployScaling()
{
  d_employScaling = true;
  s_logger->repx(logga::dbug, "employ scaling set", d_employScaling);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : initEmployAdvBasis
// ----------------------------------------------------------
//  Description  : elect to use advanced initial LP basis
//  Role         : initialization method, optional
//  Techniques   : 'd_employAdvBasis'
//  Status       : complete
// ----------------------------------------------------------
```

```
  void
  SolverIf::initEmployAdvBasis()
  {
    d_employAdvBasis = true;
    s_logger->repx(logga::dbug, "employ advanced initial bassis set", d_employAdvBasis);
  }

  // ----------------------------------------------------------
  //  MEMBER FUNCTION : initSimplexPresolver
  // ----------------------------------------------------------
  //  Description  : elect to use integer presolver
  //  Role         : initialization method, optional
  //  Techniques   : 'glp_smcp::presolve'
  //  Status       : complete
  // ----------------------------------------------------------

  void
  SolverIf::initSimplexPresolver()
  {
    d_parmSimplex.presolve = GLP_ON;         // simplex presolver
    d_simplexPresol = true;                  // used when reporting
    s_logger->repx(logga::dbug, "simplex presolver set", d_parmSimplex.presolve);
  }

  // ----------------------------------------------------------
  //  MEMBER FUNCTION : initIntegerPresolver
  // ----------------------------------------------------------
  //  Description  : elect to use integer presolver
  //  Role         : initialization method, optional
  //  Techniques   : 'glp_iocp::presolve'
  //  Status       : complete
  // ----------------------------------------------------------

  void
  SolverIf::initIntegerPresolver()
  {
    d_parmInteger.presolve = GLP_ON;         // mixed-integer presolver
    d_integerPresol = true;                  // used when reporting
    s_logger->repx(logga::dbug, "integer presolver set", d_parmInteger.presolve);
  }

  // PROBLEM BUILDING : entirely dynamic now, no ahead-of-time
  // size and klass estimates are solicited

  // ----------------------------------------------------------
  //  MEMBER FUNCTION : setProblemLabel
  // ----------------------------------------------------------
  //  Description  : relabel problem
  //  Role         : problem building
  //  Techniques   : 'glp_set_prob_name'
  //  Status       : complete
  //
  //  CAUTION: label uniqueness
  //
  //      The generated part of the label is not currently unique
  //      due to the one second resolution.  The optional
  //      supplied part should assist uniqueness.  However it may
  //      be useful to redesign this method.
  //
  // ----------------------------------------------------------

  void
  SolverIf::setProblemLabel
  (const std::string tag)                    // problem name tag
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function, tag", tag);

    // update instance data
    d_probTag = tag;

    // add or overwrite meta information
    std::string sPrefix = ::PRO;             // set at beginning of this file
    std::string sSep    = ::SEP;             // set at beginning of this file
    std::string sLabel;                      // full label

#if 0 // 1 = sleep one second to ensure name uniqueness, 0 = do nothing
    sleep(1);                                // refer <unistd.h>
#endif // 0

    // produce sTimeStamp: 00000000z0000
```

```
    // NOTE: C rather that Boost time functions were chosen for portability
    const size_t MAX = 1024;                // define buffer length
    const time_t now = time(0);             // seconds since 1970
    const tm* utc    = gmtime(&now);        // convert to tm struct in UTC time
    char timeStamp[MAX];

#if 1 // 0 = long, 1 = short
    strftime(timeStamp, MAX, "%H%M%S", utc);          // format time as string, short
#else
    strftime(timeStamp, MAX, "%Y%m%dz%H%M%S", utc);   // format time as string, long
#endif // 0

    std::string sTimeStamp(timeStamp);      // convert to std::string

    // construct name
    if ( ! sPrefix.empty() ) sLabel += sPrefix + "-";  // pro-
    sLabel                          += sTimeStamp;     // 00000000z0000
    if ( ! tag.empty() )     sLabel += sSep + tag;     // -tag

    // load
    glp_set_prob_name(d_prob, sLabel.c_str());         // GLPK API

    // update problem status
    updateProbStatus();

    // report supplied tag
    std::string sRecoverLabel = glp_get_prob_name(d_prob);
    if ( tag.empty() )
      {
        s_logger->repx(logga::dbug, "supplied problem tag", "(empty)");
      }
    else
      {
        s_logger->repx(logga::dbug, "supplied problem tag", tag);
      }

    // completion reporting
    s_logger->repx(logga::adhc, "recovered problem tag",   getProblemTag());
    s_logger->repx(logga::dbug, "recovered problem label", getProblemLabel());

  } // function 'SolverIf::setProblemLabel'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : setObjectiveSense
  // ------------------------------------------------------------
  //  Description  : set objective sense, label objective function
  //  Role         : problem building
  //  Techniques   : 'glp_set_obj_dir'
  //  Status       : complete
  // ------------------------------------------------------------

  void
  SolverIf::setObjectiveSense
  (const ObjectiveSense objSense,           // objective sense
   const std::string     tag)               // objective function name tag
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    // set private data
    d_objSense = objSense;

    // add meta information
    setObjectiveLabel(tag);

    // set objective sense
    switch ( d_objSense )
      {
      case maximize:
        glp_set_obj_dir(d_prob, GLP_MAX);
        s_logger->repx(logga::dbug, "about to call GLPK glp_set_obj_dir", GLP_MAX);
        break;
      case minimize:
        s_logger->repx(logga::dbug, "about to call GLPK glp_set_obj_dir", GLP_MIN);
        glp_set_obj_dir(d_prob, GLP_MIN);
        break;
      default:
        s_logger->repx(logga::warn, "unsupported objective sense", d_objSense);
        std::clog << "** coding error 01 in source file " << __FILE__
                  << ": unsupported objective sense" << std::endl;
        break;
```

```
    }

  // update problem status
  updateProbStatus();

  // report with recovered name and sense
  std::string sRecoverLabel = glp_get_obj_name(d_prob);
  const int sense           = glp_get_obj_dir(d_prob);
  std::string buf;
  switch ( sense )
    {
    case GLP_MIN: buf = "minimize";                                          break;
    case GLP_MAX: buf = "maximize";                                          break;
    default:
      s_logger->repx(logga::warn, "unsupported GLPK objective direction", sense);
      std::clog << "** coding error 02 in source file " << __FILE__
                << ": unsupported GLP constant" << std::endl;
      break;
    }

  s_logger->repx(logga::dbug, "objective function label", sRecoverLabel);
  s_logger->repx(logga::dbug, "objective sense set",      buf);

} // function 'SolverIf::setObjectiveSense'

// ------------------------------------------------------------
//  MEMBER FUNCTION : setObjectiveLabel
// ------------------------------------------------------------
//  Description  : label objective function
//  Role         : problem building
//  Techniques   : 'glp_set_obj_name', strips and reapplies standard label
//  Status       : complete
// ------------------------------------------------------------

void
SolverIf::setObjectiveLabel
(const std::string tag)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // create new string
  std::string buf = tag;

  // clean up 'buf'
  ::trimPrefixPlus(buf);

  // add meta information
  std::string sPrefix = ::OBJ;             // set at beginning of this file
  std::string sSep    = ::SEP;             // set at beginning of this file
  std::string sLabel;                      // full label
  if ( ! sPrefix.empty() ) sLabel += sPrefix;        // obj
  if ( ! buf.empty()     ) sLabel += sSep + buf;     // -buf
  glp_set_obj_name(d_prob, sLabel.c_str());          // GLPK API

} // function 'SolverIf::setObjectiveLabel'

// ------------------------------------------------------------
//  MEMBER FUNCTION : loadRhs
// ------------------------------------------------------------
//  Description  : load RHS value and constraint sense, label row
//  Role         : problem building
//  Techniques   : 'glp_add_rows' and 'glp_set_row_bnds'
//  Status       : complete
//
//  Design notes
//
//      The RHS can be zero and may be negative.  However a
//      strictly negative value can readily generate
//      infeasibilities and is thus flagged for further
//      consideration.
//
//  CAUTION: constraint senses {R, N} not currently supported
//
//      Constraint senses {R, N} (ranged and free) are not
//      supported here (although it would not be difficult to
//      add the code, although that would require overloading
//      'loadRhs').
//
// ------------------------------------------------------------
```

```
void
SolverIf::loadRhs
(const int             row,               // row index
 const double          rhsValue,          // RHS value
 const ConstraintSense conSense,          // constraint sense as enum
 const std::string     tag)               // constraint equation name tag
{
  // row number in string form for later reporting
  std::string sRow = indexToString(row);

  // check for negative rhsValue, this is advisory
  if ( rhsValue < 0.0 )
    {
      std::string sWarn = "negative row " + sRow + " RHS";
      s_logger->repx(logga::dbug, sWarn, rhsValue);
      incrementAlertCount();
    }

  // add row or rows as required -- comment strongly if this
  // action is non-progressive, that is, will either skip rows
  // or backfill a previously loaded row
  int nRows    = glp_get_num_rows(d_prob); // GLPK API
  int deltaRow = row - nRows - 1;          // note minus one
  if ( deltaRow == 0 )                     // progressive addition
    {
      glp_add_rows(d_prob, 1);             // GLPK API
    }
  else if ( deltaRow > 0 )                 // skipping case
    {
      glp_add_rows(d_prob, deltaRow + 1);
      s_logger->repx(logga::info, "note row " + sRow + " will skip past", deltaRow);
    }
  else                                     // backfilling case
    {
      s_logger->repx(logga::info, "note row " + sRow + " will backfill by", deltaRow);
    }

  // comment strongly if this action will overwrite an existing row
  if ( glp_get_row_name(d_prob, row) != NULL )  // overwrite case
    {
      s_logger->repx(logga::info, "note row " + sRow + " will overwrite", "");
    }

  // add meta information
  std::string sPrefix = ::CON;             // set at beginning of this file
  std::string sSep    = ::SEP;             // set at beginning of this file
  std::string sLabel;                      // full label
  if ( ! sPrefix.empty() ) sLabel += sPrefix + sRow;      // con0
  if ( ! tag.empty()     ) sLabel += sSep + tag;          // -tag
  glp_set_row_name(d_prob, row, sLabel.c_str());          // GLPK API

  // process RHS value based on constraint sense
  //
  //  * intuitive form :                    eqn <= | = | >= rhs
  //  * Numerical Recipes terminology :     M1, M3, M2
  //  * IBM OSI terminology, adopted here :  L, E, G
  //  * old GLPK terminology :              LPX_UP, LPX_FX, LPX_LO
  //  * new GLPK terminology :              GLP_UP, GLP_FX, GLP_LO
  //
  // technically, the GLPK terms refer to the associated
  // +auxiliary+ variable, when the equations are in standard
  // form -- in practice, one can view these as applying
  // directly to the constraints in ordinary form (see earlier)
  //
  // the final arguments in the GLPK calls are 'lower bound'
  // and 'upper bound'

  switch ( conSense )
    {
      // eqn <= rhs, L, GLP_UP
    case L:
      glp_set_row_bnds(d_prob, row, GLP_UP, 0.0, rhsValue);
      break;
      // eqn  = rhs, E, GLP_FX
    case E:
      glp_set_row_bnds(d_prob, row, GLP_FX, rhsValue, rhsValue); // upper bound not used
      break;
      // eqn >= rhs, G, GLP_LO
    case G:
      glp_set_row_bnds(d_prob, row, GLP_LO, rhsValue, 0.0);      // upper bound not used
      break;
```

```
      case R:
        s_logger->repx(logga::warn, "unsupported constraint sense", "R (ranged)");
        break;
      case N:
        s_logger->repx(logga::warn, "unsupported constraint sense", "N (free)");
        break;
      default:
        s_logger->repx(logga::warn, "unsupported constraint sense", conSense);
        std::clog << "** coding error 03 in source file " << __FILE__
                  << ": unsupported enum" << std::endl;
        break;
      }

    // update problem status
    updateProbStatus();

    // report with recovered name and value
    std::string sRecLabel = glp_get_row_name(d_prob, row);
    const double lb = glp_get_row_lb(d_prob, row);
    const double ub = glp_get_row_ub(d_prob, row);
    const int type  = glp_get_row_type(d_prob, row);

    // interpretation
    std::ostringstream buf;
    switch ( type )
      {
      case GLP_FR: buf << "" << " free";                                          break;
      case GLP_LO: buf << lb << " lower";                                        break;
      case GLP_UP: buf << ub << " upper";                                        break;
      case GLP_DB: buf << lb << " " << ub << " range";                           break;
      case GLP_FX: buf << lb << " fixed";                                        break;
      default:
        s_logger->repx(logga::warn, "unsupported GLPK row type", type);
        std::clog << "** coding error 04 in source file " << __FILE__
                  << ": unsupported GLP constant" << std::endl;
        break;
      }
    // often over-length so concat into just one string for
    // formatting consistency and thus readability
    const std::string msg = "load row " + sRow + " : " + sRecLabel + " " + buf.str();
    s_logger->repx(logga::dbug, msg, "");

  } // function 'SolverIf::loadRhs'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : loadObj
  // ------------------------------------------------------------
  //  Description  : load objective coefficient, label structural variable
  //  IMPORTANT    : may also add a standard non-negativity condition for that variable
  //  Role         : problem building
  //  Techniques   : 'glp_add_cols', 'glp_set_obj_coef', and 'glp_set_col_bnds'
  //  Status       : complete
  //
  //  Design notes
  //
  //      An objective coefficient may be zero or strictly
  //      negative.
  //
  //      A structural variable cannot be strictly negative under
  //      this implementation (see the caution below and check
  //      the conditional compilation).
  //
  //  CAUTION: standard non-negativity or free condition required
  //
  //      It is important to note that 'glp_add_cols' behaves as
  //      if the following API was also invoked:
  //
  //          glp_set_col_bnds(d_prob, col, GLP_FX, 0.0, 0.0)
  //
  //      Hence it is necessary to expressly call one of the
  //      following APIs (noting that unrequired bound values are
  //      ignored and can be happily set to zero):
  //
  //          glp_set_col_bnds(d_prob, col, GLP_LO, 0.0, 0.0)
  //          glp_set_col_bnds(d_prob, col, GLP_FR, 0.0, 0.0)
  //
  //      The choice of API is is controlled by conditional
  //      compilation in the code below.
  //
  // ------------------------------------------------------------
```

```
  void
  SolverIf::loadObj
  (const int         col,                    // col index (can be zero)
   const double      objValue,               // objective coefficient value
   const std::string tag)                    // objective function name tag
  {
    // check for "shift" and respond appropriately
    if ( col == 0 )
      {
        glp_set_obj_coef(d_prob, col, objValue);
        s_logger->repx(logga::dbug, "objective \"shift\" set, value", objValue);
        if ( ! tag.empty() )
          {
            s_logger->repx(logga::info, "tag string for \"shift\" ignored", tag);
          }
        return;
      }

    // create new string
    std::string buf = tag;

    // clean up 'buf'
    ::trimPrefixPlus(buf);

    // col number in string form for later reporting
    std::string sCol = indexToString(col);

    // check for zero-valued objective coefficient values and
    // issue a comment -- notwithstanding that GLPK allows
    // objective coefficients to be zero
    if ( objValue == 0.0 )                   // straight comparison, no EPS test
      {
#if 0 // 0 = suppress report, 1 = make report
        s_logger->repx(logga::adhc, "note col " + sCol + " zero-valued", "");
#endif // 0
      }

    // add column or columns as required -- comment strongly if
    // this action is non-progressive, that is, will either skip
    // cols or backfill a previously loaded col
    int nCols    = glp_get_num_cols(d_prob); // GLPK API
    int deltaCol = col - nCols - 1;          // note minus one
    if ( deltaCol == 0 )                     // progressive addition
      {
        glp_add_cols(d_prob, 1);             // GLPK API
      }
    else if ( deltaCol > 0 )                 // skipping case
      {
        glp_add_cols(d_prob, deltaCol + 1);
        s_logger->repx(logga::info, "note col " + sCol + " will skip past", deltaCol);
      }
    else                                     // backfilling case
      {
#if 0 // 0 = suppress report, 1 = make report
        s_logger->repx(logga::info, "note col " + sCol + " will backfill by", deltaCol);
#endif // 0
      }

    // comment strongly if this action will overwrite an existing col
    if ( glp_get_col_name(d_prob, col) != NULL )  // overwrite case
      {
#if 0 // 0 = suppress report, 1 = make report
        s_logger->repx(logga::info, "note col " + sCol + " will overwrite", "");
#endif
      }

    // add meta information
    std::string sPrefix = ::VAR;             // set at beginning of this file
    std::string sSep    = ::SEP;             // set at beginning of this file
    std::string sLabel;                      // full label
    if ( ! sPrefix.empty() ) sLabel += sPrefix + sCol;      // var0
    if ( ! buf.empty()     ) sLabel += sSep + buf;          // -buf
    glp_set_col_name(d_prob, col, sLabel.c_str());          // GLPK API

    // note that these bounds should be integer-valued (such that
    // val == round(val) holds) to accomodate 'markVarInteger'
    utilSetDefaultBnds(col);

    // load objective coefficient
    glp_set_obj_coef(d_prob, col, objValue);
```

```
  // update problem status
  updateProbStatus();

  // report with recovered name and value
  std::string sRecLabel = glp_get_col_name(d_prob, col);
  double      recValue  = glp_get_obj_coef(d_prob, col);
  std::ostringstream buf2;
  buf2 << recValue;                          // stream the 'double'
  const std::string msg = "load col " + sCol + " : " + sRecLabel + " " + buf2.str();
  s_logger->repx(logga::dbug, msg, "");

} // function 'SolverIf::loadObj'

// ------------------------------------------------------
//  MEMBER FUNCTION : incShift (increment "shift" term)
// ------------------------------------------------------
//  Description  : incrementally load objective constant term
//  Role         : problem building
//  Techniques   : 'glp_get_obj_coef' and 'glp_set_obj_coef'
//  Status       : complete
//
//  Design notes
//
//      This function is designed to stop multiple calls
//      clobbering earlier information.
//
//      If you do want to overwrite the current value, use
//      'loadCoef' and nominate col zero.
//
// ------------------------------------------------------

void
SolverIf::incShift
(const double constTermValue)              // objective constant term value increment
{
  // initial reporting
  static int callCount = 0;                // intended for first step debugging
  s_logger->repx(logga::adhc, "entering member function, call cnt", ++callCount);

  // CAUTION: column zero is correct in this case
  const double currentConstTermValue = glp_get_obj_coef(d_prob, 0);
  const double total = currentConstTermValue + constTermValue;
  glp_set_obj_coef(d_prob, 0, total);

  // report
  std::ostringstream buf;
  buf << currentConstTermValue << " + " << constTermValue << " = " << total;
  const std::string msg = "incr col 0 : " + buf.str();
  s_logger->repx(logga::dbug, msg, "");

} // function 'SolverIf::incShift'

// ------------------------------------------------------
//  MEMBER FUNCTION : loadCof
// ------------------------------------------------------
//  Description  : load non-zero constraint matrix coefficient
//  Role         : problem building
//  Techniques   : sparse matrix vectors, 'glp_load_matrix'
//  Status       : complete
//
//  Design notes
//
//      GLPK will accept and silently ignore constraint
//      coefficients set to zero.  However, zero values are
//      rejected in this code and are therefore never seen by
//      GLPK.
//
//      The 'glp_load_matrix' call is made each time to keep
//      the problem instance current.  This may be an expensive
//      call but I suspect not.  If need be the load statement
//      can be moved further down the formulation chain.
//
// ------------------------------------------------------

bool
SolverIf::loadCof
(const int    row,                         // row index
 const int    col,                         // col index
 const double coeffValue)                  // coefficient value
{
  // row and col numbers in string form for later reporting
```

```
    std::string sRow = indexToString(row);
    std::string sCol = indexToString(col);

    // check for a zero-valued constraint coefficient --
    // zero-valued constraint coefficients are allowed in GLPK
    // but are not stored

    if ( coeffValue == 0.0 )                // straight comparison, no EPS test
      {
        std::string sWarn =  "note row " + sRow + " col " + sCol + " coeff zero-valued";
        s_logger->repx(logga::info, sWarn, coeffValue);
        incrementAlertCount();
        return false;
      }

    // insert coefficient into the sparse matrix vectors
    d_ia.push_back(row);
    d_ja.push_back(col);
    d_ar.push_back(coeffValue);

    // load matrix (see design note above)
    const int coeffCount = d_ar.size() - s_glpkMargin;
    glp_load_matrix(d_prob, coeffCount, &d_ia[0], &d_ja[0], &d_ar[0]);     // void return

    // update problem status
    updateProbStatus();

    // report with col and name information
    std::ostringstream buf;
#if 0 // 0 = new style, 1 = old style
    buf << coeffValue;                      // stream the 'double'
    const std::string msg  = "load row " + sRow + " col " + sCol + " : " + buf.str();
#else
    buf << std::showpos << coeffValue;      // stream the 'double'
    const std::string msg  = "load cof " + sRow + " " + sCol + " : " + buf.str();
#endif // 0
    s_logger->repx(logga::dbug, msg, "");

    return true;

  } // function 'SolverIf::loadCof'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : reviseBnds
  // ------------------------------------------------------------
  //  Description  : revise variable bounds
  //  Role         : problem building
  //  Techniques   : 'glp_set_col_bnds'
  //  Status       : complete
  // ------------------------------------------------------------

  void
  SolverIf::reviseBnds
  (const int    col,                      // col index
   const double lowerValue,               // lower bound
   const double upperValue)               // upper bound
  {
    // col number in string form for later reporting
    std::string sCol = indexToString(col);

    // check col is within range and already loaded
    int nCols = glp_get_num_cols(d_prob);
    if ( col <= 0 || col > nCols )          // out of range
      {
        s_logger->repx(logga::warn, "note col " + sCol + " illegal and ignored", nCols);
        return;                             // otherwise will signal SIGABRT and exit 134
      }
    else if ( glp_get_col_name(d_prob, col) == NULL )  // not already loaded
      {
        s_logger->repx(logga::info, "note col " + sCol + " not loaded", "");
        return;                             // otherwise will signal SIGABRT and exit 134
      }

    // revise bounds
    const double inf = getInf();            // IEEE 754 (IEC 60559) infinity

    if ( lowerValue > upperValue )          // nonsensical
      {
        s_logger->repx(logga::info, "range setting faulty", "");
      }
    else if ( lowerValue == upperValue )    // equal
```

```
      {
        glp_set_col_bnds(d_prob, col, GLP_FX, lowerValue, upperValue);
      }
    else if ( lowerValue == -inf )
      {
        if ( upperValue == inf )
          {
            glp_set_col_bnds(d_prob, col, GLP_FR, 0.0, 0.0);
          }
        else                                    // upper value finite
          {
            glp_set_col_bnds(d_prob, col, GLP_UP, 0.0, upperValue);
          }
      }
    else                                        // lower value finite
      {
        if ( upperValue == inf )
          {
            glp_set_col_bnds(d_prob, col, GLP_LO, lowerValue, 0.0);
          }
        else                                    // upper value finite too
          {
            glp_set_col_bnds(d_prob, col, GLP_DB, lowerValue, upperValue);
          }
      }

    // report with recovered name and value

    //  Makhorin states "the routine 'glp_get_col_ub' returns the
    //  upper bound of 'j'-th column, that is the upper bound of
    //  corresponding structural variable -- however, if the
    //  column has no upper bound, the routine returns
    //  '+DBL_MAX'" and similar for 'glp_get_col_lb'

    const std::string sRecLabel = glp_get_col_name(d_prob, col);
    double            recLower  = glp_get_col_lb(d_prob, col);
    double            recUpper  = glp_get_col_ub(d_prob, col);

    // transform to IEEE 754 (IEC 60559) -/+inf (rather than -/+1.79769e+308)
    if ( recLower == -DBL_MAX ) recLower = -inf;
    if ( recUpper == +DBL_MAX ) recUpper = +inf;

    std::ostringstream buf1;
    std::ostringstream buf2;
    buf1 << std::showpos << recLower;           // stream the 'double'
    buf2 << std::showpos << recUpper;           // stream the 'double'
    const std::string msg
      = "load bnd " + sCol + " : " + sRecLabel + " " + buf1.str() + " " + buf2.str();
    s_logger->repx(logga::dbug, msg, "");

  } // function 'SolverIf::reviseBnds'

  // ------------------------------------------------------
  //  MEMBER FUNCTION : resetDefaultBnds
  // ------------------------------------------------------
  //  Description  : reset to pre-defined bounds
  //  Role         : problem building
  //  Techniques   : utility function 'utilSetDefaultBnds'
  //  Status       : complete
  // ------------------------------------------------------

  void
  SolverIf::resetDefaultBnds
  (const int col)
  {
    utilSetDefaultBnds(col);

  } // function 'SolverIf::resetDefaultBnds'

  // ------------------------------------------------------
  //  MEMBER FUNCTION : markVarInteger
  // ------------------------------------------------------
  //  Description  : mark structural variable as integer-valued, revises label
  //  Role         : problem building
  //  Techniques   : 'glp_set_col_kind'
  //  Status       : complete
  //
  //  Design notes
  //
  //      'GLP_IV' variables are strictly integer under GLPK,
  //      meaning they cannot additionally be binary.
```

```
//
//  Background
//
//      Robbie: MILP: Note on numerical thresholds.
//
//      This code indirectly uses the 'floor' test from
//      <cmath>.  This compares (on my system, IA32) an epsilon
//      value of 2.22045e-16.
//
//      It might be necessary to rework this code using the
//      same threshold that GLPK uses.  This test will be as
//      strict or more strict that GLPK, so acceptance here
//      will confirm the problem is okay.  However, rejection
//      may also be okay.
//
//      Andrew Makhorin, GLPK maintainer, later writes:
//
//      "Bounds of variables marked as integer must be exact
//      integer numbers with no tolerance, i.e. the condition
//      bnd == floor(bnd) must be satisfied, where bnd is a
//      lower/upper bound.
//
//      However, to check if a node solution is integer
//      feasible, [an]other criterion is used, namely, the
//      current value of a variable is considered as integral,
//      if it differs from the nearest integer at most by 1e-6
//      (this is [the] *absolute* tolerance used by default)."
//
//      See also Stephens etal (2006, section 3.4, pp 129-131)
//      on comparing floating-point numbers with bounded
//      accuracy.
//
// ------------------------------------------------------------

void
SolverIf::markVarInteger
(const int col)                             // col index
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // col number in string form for later reporting
  std::string sCol = indexToString(col);

  // check col is within range and already loaded
  int nCols = glp_get_num_cols(d_prob);
  if ( col <= 0 || col > nCols )            // out of range
    {
      s_logger->repx(logga::warn, "note col " + sCol + " illegal and ignored", nCols);
      return;                               // otherwise will signal SIGABRT and exit 134
    }
  else if ( glp_get_col_name(d_prob, col) == NULL )  // not already loaded
    {
      s_logger->repx(logga::info, "note col " + sCol + " not loaded", "");
      return;                               // otherwise will signal SIGABRT and exit 134
    }

  // overwrite meta information as appropriate
  std::string sColName = glp_get_col_name(d_prob, col);   // GLPK API
  if ( sColName.substr(0, ::len) == ::VAR ||              // first ::len chars
       sColName.substr(0, ::len) == ::VAB )
    {
      sColName.replace(0, ::len, ::VAZ);                  // replacement
      glp_set_col_name(d_prob, col, sColName.c_str());    // GLPK API
    }
  else                                      // repeat call
    {
      s_logger->repx(logga::info, "note col " + sCol + " is repeat call", "");
    }

  // check bounds are indeed integer, also a GLPK requirement
  double ub = glp_get_col_ub(d_prob, col);      // upper bound on variable
  if ( ! ::isIntegerValued(ub) )                // test for integer bound
    {
      const std::string msg = "col " + indexToString(col) + " upper bound not integer";
      s_logger->repx(logga::dbug, msg, ub);
    }
  double lb = glp_get_col_lb(d_prob, col);      // lower bound on variable
  if ( ! ::isIntegerValued(lb) )                // test for integer bound
    {
      const std::string msg = "col " + indexToString(col) + " lower bound not integer";
```

```
      s_logger->repx(logga::dbug, msg, lb);
    }

  // mark column as integer
  glp_set_col_kind(d_prob, col, GLP_IV);   // IV is integer variable

  // update problem klass
  updateProbKlass();                       // may have been 'prob_linear'

  // report
  std::string sLog = "mark col " + sCol + " : integer";
  s_logger->repx(logga::dbug, sLog, "");

} // member function 'SolverIf::markVarInteger'

// -----------------------------------------------------
//  MEMBER FUNCTION : markVarBinary
// -----------------------------------------------------
//  Description  : mark structural variable as 0-1-valued, revises label
//  Role         : problem building
//  Techniques   : 'glp_set_col_kind'
//  Status       : complete
// -----------------------------------------------------

void
SolverIf::markVarBinary
(const int col)                            // col index
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // col number in string form for later reporting
  std::string sCol = indexToString(col);

  // check col is within range and already loaded
  int nCols = glp_get_num_cols(d_prob);
  if ( col <= 0 || col > nCols )           // out of range
    {
      s_logger->repx(logga::warn, "note col " + sCol + " illegal and ignored", nCols);
      return;                              // otherwise will signal SIGABRT and exit 134
    }
  else if ( glp_get_col_name(d_prob, col) == NULL )  // not already loaded
    {
      s_logger->repx(logga::info, "note col " + sCol + " not loaded", "");
      return;                              // otherwise will signal SIGABRT and exit 134
    }

  // overwrite meta information as appropriate
  std::string sColName = glp_get_col_name(d_prob, col);   // GLPK API
  if ( sColName.substr(0, ::len) == ::VAR ||              // first ::len chars
       sColName.substr(0, ::len) == ::VAZ )
    {
      sColName.replace(0, ::len, ::VAB);                  // replacement
      glp_set_col_name(d_prob, col, sColName.c_str());    // GLPK API
    }
  else                                     // repeat call
    {
      s_logger->repx(logga::info, "note col " + sCol + " is repeat call", "");
    }

  // mark column as integer
  glp_set_col_kind(d_prob, col, GLP_BV);   // BV is binary variable

  // update problem klass
  updateProbKlass();                       // may have been 'prob_linear'

  // report
  std::string sLog = "mark col " + sCol + " : binary";
  s_logger->repx(logga::dbug, sLog, "");

} // member function 'SolverIf::markVarBinary'

// -----------------------------------------------------
//  MEMBER FUNCTION : markVarContinuous
// -----------------------------------------------------
//  Description  : mark structural variable as real-valued, revises label
//  Role         : problem building
//  Techniques   : 'glp_set_col_kind'
//  Status       : complete
// -----------------------------------------------------
```

```
void
SolverIf::markVarContinuous
(const int col)                           // col index
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // col number in string form for later reporting
  std::string sCol = indexToString(col);

  // check col is within range and already loaded
  int nCols = glp_get_num_cols(d_prob);
  if ( col <= 0 || col > nCols )          // out of range
    {
      s_logger->repx(logga::warn, "note col " + sCol + " illegal and ignored", nCols);
      return;                             // otherwise will signal SIGABRT and exit 134
    }
  else if ( glp_get_col_name(d_prob, col) == NULL )  // not already loaded
    {
      s_logger->repx(logga::info, "note col " + sCol + " not loaded", "");
      return;                             // otherwise will signal SIGABRT and exit 134
    }

  // overwrite meta information as appropriate
  std::string sColName = glp_get_col_name(d_prob, col);   // GLPK API
  if ( sColName.substr(0, ::len) == ::VAZ ||              // first ::len chars
       sColName.substr(0, ::len) == ::VAB )
    {
      sColName.replace(0, ::len, ::VAR);                  // replacement
      glp_set_col_name(d_prob, col, sColName.c_str());
    }
  else                                    // repeat call
    {
      s_logger->repx(logga::info, "note col " + sCol + " is repeat call", "");
    }

  // mark column as integer
  glp_set_col_kind(d_prob, col, GLP_CV);   // CV is continuous variable

  // reset bounds, otherwise these remain at [0,1], which seems
  // a bit counter-intuitive
  utilSetDefaultBnds(col);

  // update problem klass
  updateProbKlass();                      // may go back to 'prob_linear'

  // report
  std::string sLog = "mark col " + sCol + " : continuous";
  s_logger->repx(logga::dbug, sLog, "");

} // member function 'SolverIf::markVarContinuous'

// -----------------------------------------------------------
//  MEMBER FUNCTION : reviseObj
// -----------------------------------------------------------
//  Description  : reload objective value
//  Role         : problem building
//  Techniques   : 'glp_set_obj_coef'
//  Status       : complete
//
//  Design notes
//
//      The column tag, column kind, and column bounds remain
//      as before.  No additional checks are made.
//
// -----------------------------------------------------------

double
SolverIf::reviseObj
(const int    col,                      // col index (can be zero)
 const double objValue)                 // objective coefficient value
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code
  const double prior = glp_get_obj_coef(d_prob, col);
  glp_set_obj_coef(d_prob, col, objValue);
  return prior;

} // member function 'SolverIf::reviseObj'
```

```
  // ----------------------------------------------------------
  //  MEMBER FUNCTION : resetProblem
  // ----------------------------------------------------------
  //  Description  : empty the GLPK problem instance of external data
  //  Role         : problem building
  //  Techniques   : remove all rows and cols
  //  Status       : complete but not tested
  //
  //  Design notes
  //
  //      The old problem label remains.
  //
  //      This method does not retain internal information and
  //      thereby does not allow warm starting.
  //
  //  CAUTION: unexplained segfault bug
  //
  //      Under some circumstances, the first attempt to use
  //      'd_prob' in this function, even if just in comparison
  //      to NULL, causes a segfault.  I do not understand how
  //      this would happen, as 'd_prob' is set to NULL in the
  //      constructor member initialization list and should
  //      always be valid in straight (non-dereferenced) form.
  //
  //      In particular, 'submodel.16.guard.xem' as at commit
  //      r4554 will trigger a segfault.
  //
  // ----------------------------------------------------------

  void
  SolverIf::resetProblem()
  {
    // initial reporting
    s_logger->repx(logga::dbug, "entering member function", "");

    // defensive programming (see caution given above)
    if ( d_prob == NULL )
      {
        // cannot seem to reach here, hence the absence of
        // xeona::yeek == 25 and the use of 'logga::yeek'
        s_logger->repx(logga::yeek, "here 1, d_prob is NULL", d_prob);
      }
    else
      {
        // additional reporting as appropriate
        // YEEK 25 CODE (set by '--yeek')
        if ( xeona::yeek == 25 || xeona::yeek == 1 )
          {
            s_logger->repx(logga::dbug, "here 2, d_prob is valid", d_prob);
            s_logger->repx(logga::dbug, "here 3, will report on problem", "");
            std::ostringstream put;
            put << reportGlpkProblem();
            s_logger->putx(logga::dbug, put);
          }
      }

    const int nRows = glp_get_num_rows(d_prob);   // GLPK API
    if ( nRows > 0 )                              // protection against no rows
      {
        std::vector<int> vRows = ::seq(0, nRows); // local function, the zero is necessary
        glp_del_rows(d_prob, nRows, &vRows[0]);   // GLPK API
      }

    const int nCols = glp_get_num_cols(d_prob);
    if ( nCols > 0 )                              // protection against no cols
      {
        std::vector<int> vCols = ::seq(0, nCols);
        glp_del_cols(d_prob, nCols, &vCols[0]);
      }

#if 0 // 0 = retain GLPK parameter settings, 1 = return to default settings

    s_logger->repx(logga::dbug, "about to call GLPK lpx_reset_params", "");
    lpx_reset_parms(d_prob);

#endif // 0

    // reset data
    utilResetCoeffVectors();                // reset the three sparse matrix vectors
    utilResetObjectData();                  // reset the various data members
```

```
  // report outcome
  int rows = glp_get_num_rows(d_prob);
  int cols = glp_get_num_cols(d_prob);
  std::string info = indexToString(rows) + " x " + indexToString(cols);
  s_logger->repx(logga::info, "problem reset, rows x cols", info);

  // completion reporting (useful for debugging)
  s_logger->repx(logga::adhc, "leaving member function", "");

} // function 'SolverIf::resetProblem'

// -----------------------------------------------------------
//  MEMBER FUNCTION : recreateProblem
// -----------------------------------------------------------
//  Description  : indirectly delete and create GLPK problem instance
//  Role         : problem building, probably limited to testing
//  Techniques   : 'utilDeleteGlpkProb' and 'utilCreateGlpkProb'
//  Status       : complete
//
//  Design notes
//
//      If a new tag is supplied, it will be used.  Otherwise
//      the old label will be recycled.
//
// -----------------------------------------------------------

void
SolverIf::recreateProblem                  // delete/recreate problem instance
(std::string tag)                          // prior label recycled if tag is empty
{
  // initial reporting
  s_logger->repx(logga::dbug, "entering member function", "");

  // delete GLPK problem instance and also recover label
  const std::string label = utilDeleteGlpkProb();

  // reset object data
  utilResetCoeffVectors();
  utilResetObjectData();                   // reset the various data members

  // recreate GLPK problem instance
  if ( tag.empty() ) tag = label;          // replace with recovered label if need be
  utilCreateGlpkProb(tag);

  s_logger->repx(logga::dbug, "problem recreated", "");

} // function 'SolverIf::recreateProblem'

// METHODS INTENDED FOR TESTING

// -----------------------------------------------------------
//  MEMBER FUNCTION : convertToLP
// -----------------------------------------------------------
//  Description  : downgrade to LP if currently MILP
//  Role         : testing
//  Status       : disabled, underlying GLPK API has been removed
// -----------------------------------------------------------

void
SolverIf::convertToLP()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  s_logger->repx(logga::warn, "call not implemented", "");
}

// -----------------------------------------------------------
//  MEMBER FUNCTION : toggleObjective
// -----------------------------------------------------------
//  Description  : multiply current objective function by minus one
//  Role         : convert minimize problem to maximize (and vice-versa)
//  Comment      : provided as a work-around to the GLPK 4.25 directionality bug
//  Status       : complete
// -----------------------------------------------------------

void
SolverIf::toggleObjective()
{
  // initial reporting
```

```
    s_logger->repx(logga::dbug, "entering member function", "");

    const int nCols = glp_get_num_cols(d_prob);
    double cof   = 0;
    int    j     = 0;                             // col counter
    int    count = 0;                             // number of toggles performed
    for ( j = 0; j <= nCols; ++j )                // j = 0 is correct, includes 'shift'
      {
        cof = glp_get_obj_coef(d_prob, j);        // recover coeffient
        if ( cof != 0.0 )                         // to prevent -0's being displayed
          {
            glp_set_obj_coef(d_prob, j, -cof);    // reload
            ++count;
          }
      }
    s_logger->repx(logga::warn, "objective toggled, non-zeros", count);
  }

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : setGlpkRounding
  // ------------------------------------------------------------
  //  Description  : sets GLPK close-to-zero rounding when reporting
  //  Role         : finer control over output
  //  Techniques   : 'lpx_get_int_parm' and 'LPX_K_ROUND'
  //  Status       : complete
  //
  //  GLPK version 4.28 documentation
  //
  //      LPX_K_ROUND  type: integer, default: 0
  //                   solution rounding option:
  //                     0 - report all primal and dual values "as is"
  //                     1 - replace tiny primal and dual values by exact zero
  //
  //  CAUTION: obsolete method
  //
  //      From GLPK version 4.29, this parameter setting method
  //      was depreciated.
  //
  // ------------------------------------------------------------

  bool                                        // return previous value
  SolverIf::setGlpkRounding
  (const bool rounding)                       // 'true' = reporting uses GLPK close-to-zero
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function, rounding", rounding);

    // active code
    const int  priorInt  = lpx_get_int_parm(d_prob, LPX_K_ROUND);
    const bool priorBool = static_cast<bool>(priorInt);
    lpx_set_int_parm(d_prob, LPX_K_ROUND, static_cast<int>(rounding));

    // reporting and return
    std::ostringstream oss;
    oss << priorBool << " > " << rounding;
    s_logger->repx(logga::dbug, "close-to-zero rounding, from > to", oss.str());
    return priorBool;
  }

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : obtainGlpProbPtr
  // ------------------------------------------------------------
  //  Description  : provides direct access to 'd_prob'
  //  Role         : strictly for unit testing
  //  Techniques   : returns a 'glp_prob*'
  //  Status       : complete
  //
  //  CAUTION: dangling pointer possible
  //
  //      Pointer will become dangerous when the 'SolverIf' host
  //      is destructed.  Clients should therefore ensure the
  //      host exists before passing the pointer to a GLPK
  //      function!
  //
  // ------------------------------------------------------------

  glp_prob*
  SolverIf::obtainGlpProbPtr()
  {
#ifndef _XUTEST                               // warn as not intended for application usage
    s_logger->repx(logga::kill, "for unit testing only", "");
```

```
#endif
    if ( d_prob == NULL )                    // defensive programming
      {
        s_logger->repx(logga::warn, "returning a NULL pointer", d_prob);
      }
    return d_prob;
  }

  // PROBLEM INFORMATION : give the information as it stands,
  // calls can made before or during the problem build or before
  // or after the solver call

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : reportGlpkProblem
  // -----------------------------------------------------------
  //  Description  : wrapper to 'utilReportOnProb'
  //  Role         : problem information
  //  Status       : complete
  // -----------------------------------------------------------

  std::string
  SolverIf::reportGlpkProblem()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    return utilReportOnProb();
  }

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : getProblemLabel
  //  MEMBER FUNCTION : getProblemKlass
  //  MEMBER FUNCTION : getObjectiveSense
  //  MEMBER FUNCTION : getObjectiveSenseStr
  //  MEMBER FUNCTION : getObjectiveLabel
  //  MEMBER FUNCTION : getObjectiveTag
  //  MEMBER FUNCTION : getVarLabel
  //  MEMBER FUNCTION : getVarTag
  //  MEMBER FUNCTION : getVarCount
  //  MEMBER FUNCTION : getConCount
  //  MEMBER FUNCTION : getNonZeroCoeffCount
  // -----------------------------------------------------------
  //  Description  : get problem related information
  //  Role         : problem information
  //  Techniques   : various
  // -----------------------------------------------------------

  ProblemKlass
  SolverIf::getProblemKlass() const
  {
    return d_probKlass;                      // 'setObjectiveSense' may not yet be called
  }

  std::string
  SolverIf::getProblemLabel() const
  {
    return glp_get_prob_name(d_prob);
  }

  ObjectiveSense
  SolverIf::getObjectiveSense() const
  {
    if ( d_objSense == sense_not_specified ) // 'setObjectiveSense' not yet called
      {
        s_logger->repx(logga::info, "objective sense not specified", d_objSense);
      }
    return d_objSense;
  }

  std::string
  SolverIf::getObjectiveSenseStr() const     // interpreted
  {
    switch ( d_objSense )
      {
      case sense_not_specified: return "sense not specified";
      case minimize:            return "minimize";
      case maximize:            return "maximize";
      default:
        std::clog << "** coding error 05 in source file " << __FILE__ << std::endl;
        return "(coding error)";
      }
```

```
}

std::string                                 // with "obj-" prefix
SolverIf::getObjectiveLabel() const
{
  std::string objlabel;
  const char* cname = glp_get_obj_name(d_prob);
  if ( cname  == NULL ) objlabel = "";      // NULL is normally zero
  else                  objlabel = std::string(cname);
  return objlabel;
}

std::string                                 // without "obj-" prefix
SolverIf::getObjectiveTag() const           // tag for use in future 'SolverIf' calls
{
  std::string objtag = getObjectiveLabel();
  const std::string prefix = ::OBJ + "-";  // '::OBJ' set in this file, probably "obj"
  const int         prelen = prefix.length();   // same as: ::len + 1
  if ( objtag.substr(0, prelen) == prefix ) objtag.erase(0, prelen);
  return objtag;
}

ColumnKind
SolverIf::getVarKind
(const int col) const
{
  // CAUTION: 'glp_get_col_type' is altogether different

  const int kind = glp_get_col_kind(d_prob, col);
  switch ( kind )
    {
    case GLP_CV: return col_continuous;
    case GLP_IV: return col_integer;
    case GLP_BV: return col_binary;
    default:
      s_logger->repx(logga::warn, "unsupported GLPK col kind", kind);
      return col_other;
    }
}

std::string                                 // with "va[rzb]-" (or similar) prefix
SolverIf::getVarLabel
(const int col) const
{
  // protect against a "shift" term call (ideally such a call
  // should not be made)
  if ( col <= 0)
    {
      s_logger->repx(logga::xtra, "protecting GLPK glp_get_col_name", col);
      return "";
    }

  std::string varlabel;
  const char* cname = glp_get_col_name(d_prob, col);
  if ( cname  == NULL ) varlabel = "";      // NULL is normally zero
  else                  varlabel = std::string(cname);
  return varlabel;
}

std::string                                 // without "va[rzb]-" (or similar) prefix
SolverIf::getVarTag
(const int col) const
{
  std::string vartag     = getVarLabel(col);
  std::string prefixes[] = { ::VAR, ::VAZ, ::VAB };  // probably { "var", "vaz", "vab" }
  BOOST_FOREACH( std::string p, prefixes )
    {
      const std::string prefix = p + "-";
      const int         prelen = prefix.length();
      if ( vartag.substr(0, prelen) == prefix )
        {
          vartag.erase(0, prelen);
          break;
        }
    }
  return vartag;
}

int
SolverIf::getVarCount() const
{
```

```
  return glp_get_num_cols(d_prob);         // GLPK API
}

int
SolverIf::getConCount() const
{
  return glp_get_num_rows(d_prob);         // GLPK API
}

int
SolverIf::getNonZeroCoeffCount() const
{
  // 'glp_get_num_nz' happily returns zero if the sparse matrix
  // vectors (smv) have yet to be loaded (alternatively the
  // sparse matrix vectors may have been loaded but empty)

  int smv = d_ar.size() - s_glpkMargin;
  int glp = glp_get_num_nz(d_prob);        // GLPK API, non-zero coefficients

  if ( glp == 0 )                          // sparse matrix vectors
    {
      return smv;
    }
  else
    {
      return glp;
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : isLP
// ----------------------------------------------------------
//  Description  : reports whether the problem is LP or not
//  Role         :
//  Techniques   : various
//  Status       : complete
// ----------------------------------------------------------

bool
SolverIf::isLP() const              // all variables are continuous
{
  switch ( d_probKlass )
    {
    case prob_linear:
      return true;
    default:
      return false;
    }
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getLowerBnd
// ----------------------------------------------------------

double
SolverIf::getLowerBnd
(const int col) const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // active code -- and, if necessary, transform to IEEE 754
  // (IEC 60559) -infinity (rather than -1.79769e+308)
  double lower = glp_get_col_lb(d_prob, col);
  if ( lower == -DBL_MAX )                 // preprocessor macro, refer <cfloat>
    {
      lower = -getInf();
    }
  return lower;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getUpperBnd
// ----------------------------------------------------------

double
SolverIf::getUpperBnd
(const int col) const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");
```

```
  // active code -- and, if necessary, transform to IEEE 754
  // (IEC 60559) +infinity (rather than +1.79769e+308)
  double upper = glp_get_col_ub(d_prob, col);
  if ( upper == +DBL_MAX )                // preprocessor macro, refer <cfloat>
    {
      upper = +getInf();
    }
  return upper;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getFormattedResults
// ----------------------------------------------------------

std::string
SolverIf::getFormattedResults
(const std::string objectiveTag,
 const std::string variableTag)
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  const int tab         = 8;              // fixed indent
  const std::string sep = "  : ";         // tag-value separator

  std::ostringstream oss;
  std::ostringstream tag;

  oss << std::setw(tab) << std::right << objectiveTag << sep
      << getObjectiveValue()
      << "\n";

  const int vars = getVarCount();
  for (int var = 1; var <= vars; var++)
    {
      tag.str("");                        // empty the non-const string-stream
      tag << variableTag << var;
      oss << std::setw(tab) << std::right << tag.str() << sep
          << getVarValue(var)
          << "\n";
    }

  return oss.str();
}

// POST-BUILD MANIPULATION

// ----------------------------------------------------------
//  MEMBER FUNCTION : getObjective
// ----------------------------------------------------------
//  Description  : return current objective function as vector
//  Role         : called from 'DomainController::capset'
//  Techniques   : 'std::vector'
//  Status       : complete
//
//  Note use of 'getObjectiveTag' to recover the tag.
//
//  CAUTION: post-build
//
//      caller to ensure the call is post-build if so required!
//
//  CAUTION: shift
//
//      in GLPK, the 0th objective coefficient is the "shift"
//
// ----------------------------------------------------------

std::vector<double>                       // return current objective function
SolverIf::getObjective() const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  std::vector<double> objFunc;            // vector to load
  const int vars = getVarCount();

  // load vector
  for ( int var = 0;                      // CAUTION: zero-based index due to "shift"
        var <= vars;                      // problem cols
        var++ )
```

```
    {
      objFunc.push_back(glp_get_obj_coef(d_prob, var));
    }
  return objFunc;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : renewObjective
// ----------------------------------------------------------
//  Description  : replace objective function
//  Role         : called from 'DomainController::capset'
//  Techniques   : 'std::vector'
//  Status       : complete
//
//  CAUTION: post-build
//
//      caller to ensure the call is post-build if so required!
//
//  CAUTION: shift
//
//      in GLPK, the 0th objective coefficient is the "shift"
//
// ----------------------------------------------------------

bool
SolverIf::renewObjective
(const std::vector<double> objFunc,        // entire function
 const std::string         tag)            // objective function name tag, note default
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function, length", objFunc.size());

  // confirm length alignment
  const int objs = objFunc.size();         // supplied vector, includes the "shift"
  const int vars = getVarCount();          // current build
  if      ( objs > vars + 1 )
    {
      s_logger->repx(logga::warn, "given objective function too long", objs);
      return false;
    }
  else if ( objs < vars + 1 )
    {
      s_logger->repx(logga::warn, "given objective function too short", objs);
      return false;
    }

  // rename the objective function
  setObjectiveLabel(tag);

  // first, add the 0th coefficient, the unlabeled "shift"
  loadObj(0, objFunc.at(0));

  // then, unload the rest of the vector while simultaneously recycling existing labels
  std::string collabel;
  for ( int idx = 1;                       // one-based index, having hopped the "shift"
        idx <= vars;                       // problem cols
        idx++ )
    {
      const char* clabel = glp_get_col_name(d_prob, idx); // returns NULL if empty
      if ( clabel  == NULL ) collabel = "";
      else                   collabel = std::string(clabel);
      loadObj(idx, objFunc.at(idx), collabel);
    }
  return true;

} // function 'SolverIf::renewObjective'

// ----------------------------------------------------------
//  MEMBER FUNCTION : zeroBarMeObjective
// ----------------------------------------------------------
//  Description  : create special objective function
//  Role         : called from 'DomainController::capset'
//  Techniques   : calls 'renewObjective'
//  Status       : complete
//
//  Note also use of 'getObjectiveTag' to recover tag.
//
//  CAUTION: post-build
//
//      caller to ensure the call is post-build if so required!
//
```

```
// ----------------------------------------------------------

bool
SolverIf::zeroBarMeObjective
(const int         nonzeroGol,             // nominate the only non-zero column
 const double      value,                  // objective coefficient value
 const std::string tag)                    // new objective function name, note default
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  // confirm gol within range
  const int vars = getVarCount();
  if ( nonzeroGol > vars || nonzeroGol < 1 )    // 0th index not a sensible choice
    {
      std::ostringstream oss;
      oss << nonzeroGol << " : " << vars;
      s_logger->repx(logga::warn, "invalid gol, gol : var (col) count", oss.str());
      return false;
    }

  // create new objective function
  std::vector<double> objFunc(vars + 1, 0.0);   // fill with zeros, allow for 0th index
  objFunc.at(nonzeroGol) = value;               // overwrite the nominated col

  // call member function
  if ( ! renewObjective(objFunc, tag) )
    {
      s_logger->repx(logga::warn, "renew objective call failed", "");
      return false;
    }
  return true;

} // function 'SolverIf::zeroBarMeObjective'

// SOLVER INVOCATION

// ----------------------------------------------------------
//  MEMBER FUNCTION : runSolver
// ----------------------------------------------------------
//  Description  : public point of entry for invoking solution process
//  Role         : solver invocation, public
//  Techniques   :
//  Status       : complete
// ----------------------------------------------------------

void
SolverIf::runSolver()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // screen the problem build
  if ( d_probStatus < status_empty )       // was 'status_solvable'
    {
      std::ostringstream put;
      put << utilReportOnProb();
      s_logger->putx(logga::dbug, put);
      s_logger->repx(logga::warn, "problem incomplete and abandoning", d_probStatus);
      incrementAlertCount();
      return;                              // CAUTION: note the return
    }
  else if ( d_noise >= svif::medium )
    {
      *s_os << "\n"
            << utilReportOnProb()          // passive
            << "\n"
            << std::flush;
    }

  solverCheckProb();                       // checks integrity and sets 'd_probStatus'
  if ( d_probStatus < status_empty )       // was 'status_solvable'
    {
      s_logger->repx(logga::dbug, "abandoning before solver call", d_probStatus);
      return;
    }
  if ( d_probStatus == status_empty )
    {
      s_logger->repx(logga::rankJumpy, "empty problem (no rows and no cols)", "");
    }
```

```
    solverInvokeSolver();                   // invokes solver, set d_solverRet
    solverAnalyzeSoln();                    // sets d_solnRet and d_solnStatus

    // housekeeping
    d_solverCalls++;

    // log the status
    switch ( d_solnStatus )
      {
      case svif::soln_optimal:
        break;
      case svif::soln_feasible:
        break;
      case svif::soln_not_usable:
        s_logger->repx(logga::warn, "solution not usable, soln status", d_solnStatus);
        incrementAlertCount();
        break;
      default:
        s_logger->repx(logga::warn, "unsupported solution status", d_solnStatus);
        std::clog << "** coding error 06 in source file " << __FILE__ << std::endl;
        break;
      }

    // undertake duplicated reporting if required
    if ( d_noise >= svif::high )            // equal or above 'high'
      {
        *s_os << "\n"
              << utilReportOnProb()         // passive
              << "\n"
              << utilReportOnSolver()       // passive
              << "\n"
              << utilReportOnSoln()         // passive
              << std::flush;

        s_logger->addSmartBlank();
      }

  } // function 'SolverIf::runSolver'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : solverCheckProb (private)
  // ------------------------------------------------------------
  //  Description  : checks whether problem is fully built and acceptable to GLPK
  //  Role         : called prior to 'solverInvokeSolver'
  //  Techniques   : sets 'd_probStatus'
  //  Status       : incomplete
  //
  //  Design notes
  //
  //      This method is more comprehensive than
  //      'updateProbStatus'.
  //
  // ------------------------------------------------------------

  void
  SolverIf::solverCheckProb()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    ProblemStatus status = status_not_specified;  // initial value

    const int nVars   = glp_get_num_rows(d_prob);      // all variables
    const int nCons   = glp_get_num_cols(d_prob);      // all constraints
    const int nCoeffs = glp_get_num_nz(d_prob);        // loaded coefficients

    if ( nVars   < 1 ) s_logger->repx(logga::dbug, "insufficient cols, count",   nVars);
    if ( nCons   < 1 ) s_logger->repx(logga::dbug, "insufficient rows, count",   nCons);
    if ( nCoeffs < 1 ) s_logger->repx(logga::dbug, "insufficient coeffs, count", nCoeffs);

    if ( d_objSense == sense_not_specified )
      {
        s_logger->repx(logga::dbug, "objective sense not set, value", d_objSense);
      }

    // hunt for unfilled cols
    int colErrors = 0;
    std::string colsBad;

    for ( int j = 1; j <= nCons; j++ )
      {
```

```
        const char* clabel = glp_get_col_name(d_prob, j);
        if ( clabel == NULL )                // name not set
          {
            colErrors++;
            colsBad += " ";
            colsBad += indexToString(j);
          }
      }
    if ( ! colsBad.empty() ) colsBad.erase(0, 1); // remove leading space
    if ( colErrors )
      {
        s_logger->repx(logga::dbug, "unfilled cols found", colsBad);
      }

    // hunt for unfilled rows
    int rowErrors = 0;
    std::string rowsBad;

    for ( int i = 1; i <= nVars; ++i )
      {
        const char* clabel = glp_get_row_name(d_prob, i);
        if ( clabel == NULL )                // name not set
          {
            rowErrors++;
            rowsBad += " ";
            rowsBad += indexToString(i);
          }
      }
    if ( ! rowsBad.empty() ) rowsBad.erase(0, 1);      // remove leading space
    if ( rowErrors )
      {
        s_logger->repx(logga::dbug, "unfilled rows found", rowsBad);
      }

    // wash up (see commit r3718 for version which rejects empty problems)
    if      ( nVars   == 0                      &&  // no structural variables
              nCons   == 0                      &&  // no constraints
              nCoeffs == 0                      &&  // no constraint matrix entries
              d_objSense > sense_not_specified  &&  // objective sense not explicity set
              colErrors == 0                    &&  // unfilled cols
              rowErrors == 0 )                      // unfilled rows
      {
        status = status_empty;
      }
    else if ( nVars   < 1                       ||  // no structural variables
              nCons   < 1                       ||  // no constraints
              nCoeffs < 1                       ||  // no constraint matrix entries
              d_objSense == sense_not_specified ||  // objective sense not explicity set
              colErrors > 0                     ||  // unfilled cols
              rowErrors > 0 )                       // unfilled rows
      {
        status = status_incomplete;
      }
    else
      {
        status = status_solvable;
      }
    d_probStatus = status;

  } // function 'SolverIf::solverCheckProb'

  // ------------------------------------------------------
  //  MEMBER FUNCTION : solverInvokeSolver (private)
  // ------------------------------------------------------
  //  Description  : match solver to klass, invoke solver, and check exit status
  //  Role         : solver invocation
  //  Techniques   :
  //  Status       : complete
  //
  //  Design notes
  //
  //      Invoke solver and check the solver exit status.  The
  //      matching of solver type to problem klass is also
  //      undertaken.
  //
  //      The logic in this function is quite involved and users
  //      should confirm that it meets their needs.
  //
  //      A problem which is MILP will override any preference to
  //      use the interior point solver.
  //
```

```
//      Note that the solution is analyzed later.  At this
//      point, only solver issues are checked.  This means that
//      solution issues like feasibility and optimality, are
//      NOT examined here.
//
//      Solver success returns now vary!  LPX_E_OK is 200,
//      while success for a glp-prefixed call is simply zero.
//
//      The actual solver calls are "wrapped" in s_os newline
//      and flush calls.
//
//   'glpsol' API call logic (for comparison)
//
//      'glpsol' is the standalone command-line GLPK solver.
//
//      'glpsol' contains a similar API call logic to that used
//      below -- for GLPK 4.42, refer to:
//
//          source   : src/glpapi20.c
//          line     : 1033
//          function : 'glp_main'
//          search   : "/* solve the problem */"
//
//      This note was added long after the 'xeona' solver
//      interface was coded -- nonetheless this information
//      could be useful if there is a need to fix or refine the
//      API call logic used here.
//
//      In particular, calls to 'glp_set_bfcp' -- change basis
//      factorization control parameters -- might be required.
//      That said, the control settings employed appear rather
//      arcane -- hence the code here simply accepts the
//      defaults.
//
//      Another issue is that 'glp_intopt' is smarter than the
//      original MIP call logic used here.  That said, the
//      previous "two call" logic has been retained -- for no
//      other reason than it works.
//
// ------------------------------------------------------------

void
SolverIf::solverInvokeSolver()
{
  // initial reporting
  s_logger->repx(logga::xtra, "entering member function", "");

  // set local trip level
  const logga::Rank trip = logga::dbug;

  // update problem klass and status
  updateProbKlass();
  updateProbStatus();

  // ONE: scale the problem as appropriate

  const int scaleFlag = GLP_SF_AUTO;       // see GLPK manual for more options

  switch ( d_employScaling )
    {
    case true:
      s_logger->repx(logga::dbug, "GLPK glp_scale_prob call", scaleFlag);
      s_logger->addDumbBlank(trip);
      if ( ! logga::checkReportLevel(trip) ) glp_term_out(GLP_OFF);
      glp_scale_prob(d_prob, scaleFlag);
      if ( ! logga::checkReportLevel(trip) ) glp_term_out(GLP_ON);
      s_logger->addDumbBlank(trip);
      break;
    case false:
      s_logger->repx(logga::adhc, "GLPK problem scaling omitted", "");
      break;
    }

  // TWO: construct the basis as appropriate

  const int basisFlag = 0;                 // only 0 is currently supported

  switch ( d_employAdvBasis )
    {
    case true:
      s_logger->repx(logga::dbug, "GLPK glp_adv_basis call", basisFlag);
```

```
      s_logger->addDumbBlank(trip);
      if ( ! logga::checkReportLevel(trip) ) glp_term_out(GLP_OFF);
      glp_adv_basis(d_prob, basisFlag);
      if ( ! logga::checkReportLevel(trip) ) glp_term_out(GLP_ON);
      s_logger->addDumbBlank(trip);
      break;
    case false:
      s_logger->repx(logga::adhc, "GLPK construct adv basis omitted", "");
      break;
    }

  // THREE: invoke solver based on problem klass and any client
  //        code preferences regarding solver type

  switch ( d_probKlass )
    {
      // linear problem
    case prob_linear:
      switch ( d_solverType )
        {
        case solver_other:                          // no preference from client code
        case solver_simplex:                        // elected
        case solver_integer:                        // cannot be explicitly elected
          s_logger->repx(logga::dbug, "GLPK glp_simplex call", "simplex solver");
          d_solverType = solver_simplex;            // to enable correct reporting
          s_logger->addDumbBlank(trip);
          d_solverRet  = glp_simplex(d_prob,        // simplex solver call
                                     &d_parmSimplex);
          s_logger->addDumbBlank(trip);
          break;
        case solver_interior:                       // elected
          s_logger->repx(logga::dbug, "GLPK glp_interior call","interior point solver");
          d_solverType = solver_interior;           // to enable correct reporting
          s_logger->addDumbBlank(trip);
          d_solverRet  = glp_interior(d_prob,       // interior point solver call
                                      &d_parmInterior);
          s_logger->addDumbBlank(trip);
          break;
        default:
          s_logger->repx(logga::warn, "unsupported solver type", d_solverType);
          std::clog << "** coding error 07 in source file " << __FILE__ << std::endl;
          break;
        } // solver type code
      // report on solver return
      if ( d_solverRet == 0 )                       // covers all solvers
        {
          s_logger->repx(logga::dbug,
                         "main solver success, solver return",
                         d_solverRet);
        }
      else
        {
          s_logger->repx(logga::warn,
                         "main solver FAIL, solver return",
                         d_solverRet);
        }
      break; // linear problem

      // non-linear problem
    case prob_mixed_integer:
    case prob_mixed_zero_one:
    case prob_pure_integer:
    case prob_pure_zero_one:
      s_logger->repx(logga::dbug, "GLPK glp_simplex call", "simplex solver first");
      d_solverType = solver_simplex;                // to enable correct reporting
      s_logger->addDumbBlank(trip);
      d_solverRet  = glp_simplex(d_prob,            // simplex solver call FIRST
                                 &d_parmSimplex);
      s_logger->addDumbBlank(trip);
      if ( d_solverRet == 0 )                       // check return
        {
          s_logger->repx(logga::dbug, "GLPK glp_intopt call", "integer solver second");
          d_solverType = solver_integer;            // to enable correct reporting
          s_logger->addDumbBlank(trip);
          d_solverRet  = glp_intopt(d_prob,         // integer solver call SECOND
                                    &d_parmInteger);
          s_logger->addDumbBlank(trip);
          if ( d_solverRet == 0 )
            {
              s_logger->repx(logga::dbug,
                             "MIP solver success, solver return",
```

```
                                d_solverRet);
              }
            else
              {
                s_logger->repx(logga::dbug,
                               "MIP solver FAIL, solver return",
                               d_solverRet);
              }
          }
        else
          {
            s_logger->repx(logga::dbug,
                           "LP relaxation FAIL, solver return",
                           d_solverRet);
          }
        break; // non-linear problem

      default:
        s_logger->repx(logga::warn, "unsupported problem klass", d_probKlass);
        std::clog << "** coding error 08 in source file " << __FILE__ << std::endl;
        break;
      } // problem klass

    // revise problem status
    d_probStatus = status_submitted;         // moreover, a solver must have run

  } // function 'SolverIf::solverInvokeSolver'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : solverAnalyzeSoln (private)
  // ------------------------------------------------------------
  //  Description  :
  //  Role         : solver invocation
  //  Techniques   :
  //  Status       : complete
  // ------------------------------------------------------------

  void
  SolverIf::solverAnalyzeSoln()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    // obtain solution status
    switch ( d_solverType )
      {
      case solver_simplex:
        d_solnRet = glp_get_status(d_prob);
        break;
      case solver_interior:
        d_solnRet = glp_ipt_status(d_prob);
        break;
      case solver_integer:
        d_solnRet = glp_mip_status(d_prob);
        break;
      default:
        s_logger->repx(logga::warn, "unsupported solver type", d_solverType);
        std::clog << "** coding error 09 in source file " << __FILE__ << std::endl;
        break;
      }

    // process solution status (based on assumption that GLPK
    // encapsulates the solver status in these returns)

    switch ( d_solverType )
      {
      case solver_simplex:
        switch ( d_solnRet )
          {
          case GLP_OPT:
            d_solnStatus = soln_optimal;
            break;
          case GLP_FEAS:
            d_solnStatus = soln_feasible;
            break;
          case GLP_INFEAS:
          case GLP_NOFEAS:
          case GLP_UNBND:
          case GLP_UNDEF:
            d_solnStatus = soln_not_usable;
            break;
```

```
          default:
            s_logger->repx(logga::warn, "unsupported GLPK solver return", d_solnRet);
            std::clog << "** coding error 10 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      case solver_interior:
        switch ( d_solnRet )
          {
          case GLP_OPT:
            d_solnStatus = soln_optimal;
            break;
          case GLP_UNDEF:
            d_solnStatus = soln_not_usable;
            break;
          default:
            s_logger->repx(logga::warn, "unsupported GLPK solver return", d_solnRet);
            std::clog << "** coding error 11 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      case solver_integer:
        switch ( d_solnRet )
          {
          case GLP_OPT:
            d_solnStatus = soln_optimal;
            break;
          case GLP_FEAS:
            d_solnStatus = soln_feasible;
            break;
          case GLP_NOFEAS:
          case GLP_UNDEF:
            d_solnStatus = soln_not_usable;
            break;
          default:
            std::clog << "** coding error 12 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      default:
        std::clog << "** coding error 13 in source file " << __FILE__ << std::endl;
        break;
      }

  } // function 'SolverIf::solverAnalyzeSoln'

  // SOLUTION RECOVERY : usage must follow solver call

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : getObjectiveValue
  // -----------------------------------------------------------
  //  Description  : return final objective value, which may or may not be optimal
  //  Role         : solution recovery
  //  Techniques   : 'glp_get_obj_val' and similar
  //  Status       : complete
  // -----------------------------------------------------------

  double
  SolverIf::getObjectiveValue() const
  {
    // note out of order call
    if ( d_probStatus < status_submitted )
      s_logger->repx(logga::warn, "problem not submitted to solver", "");

    // process call based on solver type
    double objective = getNaN();
    switch ( d_solverType )
      {
      case solver_simplex:
        objective = glp_get_obj_val(d_prob); // also valid part way thru solution process
        break;
      case solver_interior:
        objective = glp_ipt_obj_val(d_prob);
        break;
      case solver_integer:
        objective = glp_mip_obj_val(d_prob);
        break;
      default:
        std::clog << "** coding error 14 in source file " << __FILE__ << std::endl;
        break;
      }
```

```
  return objective;

} // function 'SolverIf::getObjectiveValue'

// ----------------------------------------------------------
//  MEMBER FUNCTION : getVarValue
// ----------------------------------------------------------
//  Description  : return final structural var value, which may or may not be optimal
//  Role         : solution recovery
//  Techniques   : 'glp_get_col_val' and similar
//  Status       : complete
// ----------------------------------------------------------

double
SolverIf::getVarValue
(const int col) const
{
  // note out of order call
  if ( d_probStatus < status_submitted )
    s_logger->repx(logga::warn, "problem not submitted to solver", "");

  // process call based on solver type
  double value = getNaN();
  switch ( d_solverType )
    {
    case solver_simplex:
      value = glp_get_col_prim(d_prob, col);
      break;
    case solver_interior:
      value = glp_ipt_col_prim(d_prob, col);
      break;
    case solver_integer:
      value = glp_mip_col_val(d_prob, col);
      break;
    default:
      std::clog << "** coding error 15 in source file " << __FILE__ << std::endl;
      break;
    }

  return value;

} // function 'SolverIf::getVarValue'

// ----------------------------------------------------------
//  MEMBER FUNCTION : getIntegerVarValue
// ----------------------------------------------------------
//  Description  : return final integer var value, which may or may not be optimal
//  Role         : solution recovery
//  Techniques   : 'glp_get_col_val' and similar
//  Status       : complete
//
//  Design notes
//
//      See Becker (2007, ch12) regarding floating-point
//      exception code.  This code complies with TR1 and C99.
//
// ----------------------------------------------------------

int
SolverIf::getIntegerVarValue
(const int col) const
{
  // check if binary
  const int colkind = glp_get_col_kind(d_prob, col);
  if ( colkind != GLP_IV )
    {
      s_logger->repx(logga::warn, "value not integer, GLPK col kind", colkind);
    }

  // recover value
  const double value = getVarValue(col);

  // convert to integer (some elements may move to 'std::tr1::' under <cfenv>)
  fexcept_t feStatus;
  const int feFlag = FE_INEXACT;
  fegetexceptflag(&feStatus, feFlag);      // obtain current floating-point status
  feclearexcept(feFlag);                   // duly clear status
  const double dint = rint(value);         // refer <cmath>
  if ( fetestexcept(feFlag) )              // check for changed status
    {
```

```
      std::ostringstream oss;
      oss << value << " : " << dint;
      s_logger->repx(logga::warn, "inexact integer conversion", oss.str());
    }
  fesetexceptflag(&feStatus, feFlag);      // reinstate floating-point status

  // cast to integer and return
  const int iint = static_cast<int>(dint);
  return iint;

} // function 'SolverIf::getIntegerVarValue'

// ------------------------------------------------------
//  MEMBER FUNCTION : getBinaryVarValue
// ------------------------------------------------------
//  Description  : return final binary var value, which may or may not be optimal
//  Role         : solution recovery
//  Techniques   : 'glp_get_col_val' and similar
//  Status       : complete
// ------------------------------------------------------

bool                                       // 'false' = 0 and 'true' = 1 (like C++)
SolverIf::getBinaryVarValue
(const int col) const
{
  // check if binary
  const int colkind = glp_get_col_kind(d_prob, col);
  if ( colkind != GLP_BV )
    {
      s_logger->repx(logga::warn, "value not binary, GLPK col kind", colkind);
    }

  // process, noting both zero and unity are accurately represented
  const double value = getVarValue(col);
  if      ( value == 0.0 ) return false;
  else if ( value == 1.0 ) return true;
  else
    {
      s_logger->repx(logga::warn, "problem identifying binary", "");
      return true;                         // need something (or else go to 'tribool')
    }

} // function 'SolverIf::getBinaryVarValue'

// ------------------------------------------------------
//  MEMBER FUNCTION : getObjCoeff
// ------------------------------------------------------
//  Description  : get objective coefficient
//  Role         : solution recovery
//  Techniques   : 'glp_get_obj_coeff'
//  Status       : complete
// ------------------------------------------------------

double                                     // simple retrieval
SolverIf::getObjCoeff                      // an objective coefficient value
(const int col) const
{
  return glp_get_obj_coef(d_prob, col);    // col = 0 is legal and returns 'shift'
}

// ------------------------------------------------------
//  MEMBER FUNCTION : getObjCoeffXVarValue
// ------------------------------------------------------
//  Description  : get objective coefficient x variable value
//  Role         : solution recovery
//  Techniques   : local calls
//  Status       : complete
// ------------------------------------------------------

double
SolverIf::getObjCoeffXVarValue             // getObjCoeff x getVarValue
(const int col) const
{
  // note out of order call
  if ( d_probStatus < status_submitted )
    {
      s_logger->repx(logga::warn, "problem not submitted to solver", "");
    }

  return getObjCoeff(col) * getVarValue(col);
```

```
} // function 'SolverIf::getObjCoeffXVarValue'

// -----------------------------------------------------------
//  MEMBER FUNCTION : getSlackValue
// -----------------------------------------------------------
//  Description  : get LP slack value
//  Role         : solution recovery
//  Techniques   :
//  Status       : complete, needs confirmation
// -----------------------------------------------------------

double
SolverIf::getSlackValue
(const int row) const
{
  // complain if out of order call
  if ( d_probStatus < status_submitted )
    s_logger->repx(logga::warn, "problem not submitted to solver", "");

  // process call based on solver type
  switch ( d_solverType )
    {
    case solver_simplex:
    case solver_integer:                        // LP relaxation
      return glp_get_row_dual(d_prob, row);     // reduced cost of auxiliary variable
      break;
    case solver_interior:
      return glp_ipt_row_dual(d_prob, row);     // reduced cost of axillary variable
      break;
    default:
      return getNaN();
      break;
    }

} // function 'SolverIf::getSlackValue'

// -----------------------------------------------------------
//  MEMBER FUNCTION : getShadowValue
// -----------------------------------------------------------
//  Description  : get LP shadow value
//  Role         : solution recovery
//  Techniques   :
//  Status       : complete, needs confirmation
// -----------------------------------------------------------

double
SolverIf::getShadowValue
(const int col) const
{
  // complain if out of order call
  if ( d_probStatus < status_submitted )
    s_logger->repx(logga::warn, "problem not submitted to solver", "");

  // process call based on solver type
  switch ( d_solverType )
    {
    case solver_simplex:
    case solver_integer:                        // LP relaxation
      return glp_get_col_dual(d_prob, col);     // reduced cost of structural variable
      break;
    case solver_interior:
      return glp_ipt_col_dual(d_prob, col);     // reduced cost of structural variable
      break;
    default:
      return getNaN();
      break;
    }

} // function 'SolverIf::getShadowValue'

// -----------------------------------------------------------
//  MEMBER FUNCTION : isUsableSoln
// -----------------------------------------------------------
//  Description  : return true or false based on solution status
//  Role         :
//  Techniques   :
//  Status       : complete
// -----------------------------------------------------------

bool
SolverIf::isUsableSoln() const             // feasible plus other usefulness tests
```

```
{
  // note out of order call
  if ( d_probStatus < status_submitted )
    {
      s_logger->repx(logga::warn, "problem not submitted to solver", "");
    }

  switch ( d_solnStatus )
    {
    case soln_optimal:           // solution optimal
    case soln_feasible:          // solution usable but not optimal
      return true;

    case soln_not_usable:        // solution not usable but problem not proven bad
    case soln_proven_bad:        // problem proven bad
      return false;

    case soln_faulty:
    case soln_other:
      return false;

    case soln_undefined:
      s_logger->repx(logga::warn, "undefined solution, soln status", d_solnStatus);
      return false;

    default:
      std::clog << "** coding error 16 in source file " << __FILE__ << std::endl;
      return false;
    }

} // function 'SolverIf::isUsableSoln'

// ------------------------------------------------------------
//  MEMBER FUNCTION : isOptimalSoln
// ------------------------------------------------------------
//  Description  : return true or false based on solution status
//  Role         :
//  Techniques   :
//  Status       : complete
// ------------------------------------------------------------

bool
SolverIf::isOptimalSoln() const            // proven optimal
{
  // note out of order call
  if ( d_probStatus < status_submitted )
    s_logger->repx(logga::warn, "problem not submitted to solver", "");

  switch ( d_solnStatus )
    {
    case soln_optimal:
      return true;
    default:
      return false;
    }

} // function 'SolverIf::isOptimalSoln'

// ------------------------------------------------------------
//  MEMBER FUNCTION : writeInfo
// ------------------------------------------------------------
//  Description  : invokes appropriate GLPK print calls
//  Role         : diagnostics
//  Techniques   : 'lxp_print_*'
//  Status       : complete
//
//  CAUTION: file overwrite
//
//      This function will overwrite any previous files.
//
//  CAUTION: spaces to underscores
//
//      This function will convert any space characters (' ')
//      in the supplied filestem to underscore characters ('_').
//
// ------------------------------------------------------------

void
SolverIf::writeInfo                        // wrapper to std::string function
(const char* filestem) const               // filestem name, extensions added later
{
```

```
  if ( filestem == NULL )                  // protect against NULL char*
    {
      writeInfo("");
    }
  else
    {
      const std::string buf(filestem);     // C-string conversion
      writeInfo(buf);                      // simple wrapper to the std::string variant
    }
}

void
SolverIf::writeInfo() const
{
  const std::string label = getProblemLabel();
  writeInfo(label);
}

void
SolverIf::writeInfo                        // workhorse
(const std::string filestem) const
{
  // note out of order call
  if ( d_probStatus < status_submitted )
    s_logger->repx(logga::warn, "problem not submitted to solver", "");

  // process filestem
  std::string sFilestem = filestem;                             // remove const
  std::replace(sFilestem.begin(), sFilestem.end(), ' ', '_');  // spaces to underscores

  // concatenate extensions
  std::string sProbFilename = sFilestem + ".prob";
  std::string sGlpkFilename = sFilestem + ".glpk";
  std::string sSolnFilename = sFilestem + ".soln";
  std::string sSensFilename = sFilestem + ".sens";

  // convert to non-const char* for GLPK interface
  char* cProbFilename = const_cast<char*>(sProbFilename.c_str());
  char* cGlpkFilename = const_cast<char*>(sGlpkFilename.c_str());
  char* cSolnFilename = const_cast<char*>(sSolnFilename.c_str());
  char* cSensFilename = const_cast<char*>(sSensFilename.c_str());

  *s_os << "\n" << std::flush;             // for tidy output

  // call various GLPK print utilities, intended for visual analysis
  glp_write_lp(d_prob, NULL, cProbFilename);    // problem details in CPLEX format
  glp_write_prob(d_prob, 0, cGlpkFilename);     // problem details in GLPK format
  xeona::readonly(cProbFilename);
  xeona::readonly(cGlpkFilename);

  if ( d_solnRet != 0 )     // status routines have returned a three-digit int
    {
      switch ( d_solverType )
        {
        case solver_simplex:
          glp_print_sol(d_prob, cSolnFilename);                 // solution details
          glp_print_ranges(d_prob, 0, NULL, 0, cSensFilename);  // 'lpx_print_sens_bnds'
          xeona::readonly(cSolnFilename);
          xeona::readonly(cSensFilename);
          break;
        case solver_interior:
          glp_print_ipt(d_prob, cSolnFilename);           // solution details
          xeona::readonly(cSolnFilename);
          break;
        case solver_integer:
          glp_print_mip(d_prob, cSolnFilename);           // solution details
          xeona::readonly(cSolnFilename);
          break;
        default:
          std::clog << "** coding error 17 in source file " << __FILE__ << std::endl;
          break;
        }
    }

  // some reporting, readily middle-button mouse pasted onto the command-line
  *s_os << "\n"
        << "to view output: cat "
        << sFilestem << ".{prob,soln,sens}"
        << " | less"
        << "\n"
        << std::flush;
```

```
} // function 'SolverIf::writeInfo'

// ----------------------------------------------------------
//  MEMBER FUNCTION : yesOrNo
// ----------------------------------------------------------
//  Description  : gives "yes" or "no"
//  Role         : used to reinterpret bool-returning calls
//  Status       : complete
// ----------------------------------------------------------

std::string
SolverIf::yesOrNo
(const bool trueOrFalse) const
{
  return trueOrFalse ? "yes" : "no";
}

// MISCELLANEOUS

// ----------------------------------------------------------
//  MEMBER FUNCTION : getInf
//  MEMBER FUNCTION : getNaN
// ----------------------------------------------------------
//  Description  : return non-a-number (quiet form)
//  Role         : miscellaneous
//  Techniques   : <limits>
//  Status       : complete
// ----------------------------------------------------------

double
SolverIf::getInf() const
{
  return std::numeric_limits<double>::infinity();    // <limits>
}

double
SolverIf::getNaN() const
{
  return std::numeric_limits<double>::quiet_NaN();   // <limits>
}

// UTILITY FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : utilCreateGlpkProb (private)
// ----------------------------------------------------------
//  Description  : create GLPK problem instance and add label
//  Role         : utility function
//  Techniques   :
//  Status       : complete
// ----------------------------------------------------------

void
SolverIf::utilCreateGlpkProb
(const std::string tag)
{
  // create empty GLPL problem instance
  s_logger->repx(logga::dbug, "about to call GLPK glp_create_prob", "");
  d_prob = glp_create_prob();

  // confirm empty GLPK problem instance
  if ( d_prob == NULL )
    {
      s_logger->repx(logga::warn, "GLPK glp_create_problem gave NULL", d_prob);
      incrementAlertCount();
      return;                              // CAUTION: note early return
    }

  // label the problem
  setProblemLabel(tag);

  // update problem klass
  updateProbKlass();                       // an empty problem is deemed linear!

  // update problem status
  updateProbStatus();

  // update instance count
  s_instanceCount++;
  s_logger->repx(logga::xtra, "SolverIf and glp_prob instances", s_instanceCount);
```

```
} // function 'SolverIf::utilCreateGlpkProb'

// -----------------------------------------------------------
//  MEMBER FUNCTION : utilSetGlpkDefaults (private)
// -----------------------------------------------------------
//  Description  : set construction-time defaults
//  Role         : utility function
//  Techniques   : 'glp_*cp' structs
//  Status       : complete
//
//  Design notes
//
//      GLPK parameters are initially set here.
//
//      These values could also be reset in the client code if
//      suitable public member functions are written.
//
//      See the GLPK manual for details.
//
// -----------------------------------------------------------

void
SolverIf::utilSetGlpkDefaults()
{
  // determine appropriate message level from 'd_noise' setting
  int msglev = GLP_MSG_ALL;               // set high
  switch ( d_noise )                      // set in constructor
    {
    case svif::not_specified:             // not overridden, a coding error
      std::clog << "** coding error 18 in source file " << __FILE__ << std::endl;
      break;
    case svif::silent:
      msglev = GLP_MSG_OFF;               // no output (but leakage may occur)
      break;
    case svif::low:
      msglev = GLP_MSG_ERR;               // errors and warnings
      break;
    case svif::medium:
      msglev = GLP_MSG_ON;                // normal
      break;
    case svif::high:
      msglev = GLP_MSG_ALL;               // normal plus information
      break;
    default:
      std::clog << "** coding error 19 in source file " << __FILE__ << std::endl;
    }

  // determine time limit in milliseconds
  int tmlim;
  tmlim                    = 500 * 1000;   // 500s = 08:20:00
  tmlim                    = INT_MAX;      // 2147483647ms = 2147483s = 596:31:23 ~= 24d
  if ( svif::DBUG ) tmlim  = INT_MAX;      // user modifiable

  // SIMPLEX CONTROL PARAMETERS

  d_parmSimplex.msg_lev    = msglev;       // message level, see above
  d_parmSimplex.tm_lim     = tmlim;        // searching time limit in milliseconds
  d_parmSimplex.out_frq    = 100;          // output frequency in iterations (200) [1]
  d_parmSimplex.out_dly    = 0;            // output delay in milliseconds (0) [2]
  d_parmSimplex.presolve   = GLP_OFF;      // LP presolver (GLP_OFF)

  d_parmSimplex.meth       = GLP_PRIMAL;   // simplex method option (GLP_PRIMAL)
  d_parmSimplex.pricing    = GLP_PT_PSE;   // pricing technique (GLP_PT_PSE)
  d_parmSimplex.r_test     = GLP_RT_HAR;   // ratio test technique (GLP_RT_HAR)
  d_parmSimplex.it_lim     = INT_MAX;      // simplex iteration limit (INT_MAX)

  // INTERIOR POINT CONTROL PARAMETERS

  d_parmInterior.msg_lev   = msglev;       // message level, see above
  d_parmInterior.ord_alg   = GLP_ORD_AMD;  // ordering algorithm prior to Cholesky

  // MIXED-INTEGER CONTROL PARAMETERS

  d_parmInteger.msg_lev    = msglev;       // message level, see above
  d_parmInteger.tm_lim     = tmlim;        // searching time limit in milliseconds
  d_parmInteger.out_frq    = 1000;         // output frequency in milliseconds (5000)[3]
  d_parmInteger.out_dly    = 0;            // output delay in milliseconds (10000) [2]
  d_parmInteger.presolve   = GLP_OFF;      // mixed-integer presolver (GLP_OFF)

  d_parmInteger.br_tech    = GLP_BR_DTH;   // branching technique
```

```
  d_parmInteger.bt_tech    = GLP_BT_BLB;   // backtracking technique
  d_parmInteger.pp_tech    = GLP_PP_ALL;   // preprocessing technique
  d_parmInteger.fp_heur    = GLP_OFF;      // feasibility pump heuristic technique
  d_parmInteger.gmi_cuts   = GLP_OFF;      // mixed-integer cut technique
  d_parmInteger.mir_cuts   = GLP_OFF;      // mixed-integer rounding technique
  d_parmInteger.cov_cuts   = GLP_OFF;      // enable/disable mixed cover cuts
  d_parmInteger.clq_cuts   = GLP_OFF;      // enable/disable generating clique cuts
  d_parmInteger.br_tech    = GLP_BR_DTH;   // branching technique
  d_parmInteger.binarize   = GLP_OFF;      // binarization option [4]

  // GLPK 4.41 defaults sometimes recorded in round brackets
  //
  // [1] output frequency in iterations : this parameter
  // specifies how frequently the solver sends information
  // about the solution process to the terminal
  //
  // [2] output delay in milliseconds : this parameter
  // specifies how long the solver should delay sending
  // information about the solution process to the terminal
  //
  // [3] output frequency in milliseconds : this parameter
  // specifies how frequently the solver sends information
  // about the solution process to the terminal
  //
  // [4] binarization option (used only if the presolver is
  // enabled) : replace general integer variables by binary
  // ones, do not use binarization
  //
  // (source: GLPK 4.41 manual)

  // reporting preparation
  const int decimalPlaces = 1;
  std::ostringstream oss;
  oss << std::fixed << std::setprecision(decimalPlaces)
      << (static_cast<double>(tmlim) / 1000.0);
  const std::string tmlimSeconds = oss.str();

  // report
  s_logger->repx(logga::xtra, "GLPK msg_lev message level",    msglev);
  s_logger->repx(logga::xtra, "GLPK tm_lim timeout (seconds)", tmlimSeconds);

} // function 'SolverIf::utilSetGlpkDefaults'

// ----------------------------------------------------------
//  MEMBER FUNCTION : utilDeleteGlpkProb (private)
// ----------------------------------------------------------
//  Description  : delete GLPK problem instance, recover label
//  Role         : utility function, supports the destructor and reset
//  Techniques   :
//  Status       : complete
// ----------------------------------------------------------

const std::string
SolverIf::utilDeleteGlpkProb()
{
  // recover label
  std::string label;                       // empty string

  // free GLPK problem instance memory -- but not GLPK library environment memory
  if ( d_prob != NULL )
    {
      s_logger->repx(logga::dbug, "about to call GLPK glp_delete_prob", "");
      label = glp_get_prob_name(d_prob);   // returns const char*
      glp_delete_prob(d_prob);             // also release GLPK problem instance memory
      d_prob = NULL;                       // null-out the pointer
      s_instanceCount--;                   // update instance count
    }
  else
    {
      s_logger->repx(logga::warn, "problem instance unexpectedly NULL", d_prob);
    }

  // return recovered label
  return label;

} // function 'SolverIf::utilDeleteGlpkProb'

// ----------------------------------------------------------
//  MEMBER FUNCTION : utilSetDefaultBnds
// ----------------------------------------------------------
//  Description  :
```

```
  //  Role         :
  //  Techniques   :
  //  Status       : complete
  // ------------------------------------------------------

  void
  SolverIf::utilSetDefaultBnds
  (int col)
  {

#if   (XE_BNDS_DEFAULT == 0)

    // apply the free non-constraint -- which might also
    // generate "PROBLEM HAS UNBOUNDED SOLUTION" output from GLPK

    glp_set_col_bnds(d_prob, col, GLP_FR, 0.0, 0.0);   // zero bounds okay for FR

#elif (XE_BNDS_DEFAULT == 1)

    // apply the standard non-negativity constraint for a structural variable
    //
    //  * hence :                       x >= 0
    //  * in GLPK terminology :         GLP_LO
    //
    // the final arguments in the GLPK call are 'lower bound' and 'upper bound'

    glp_set_col_bnds(d_prob, col, GLP_LO, 0.0, 0.0);   // zero upper bound okay for LO

#endif // XE_BNDS_DEFAULT

  } // function 'SolverIf::utilSetDefaultBnds'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : utilResetCoeffVectors (private)
  // ------------------------------------------------------------
  //  Description  : reset vectors using 'clear' and then 'resize'
  //  Role         : utility
  //  Status       : complete
  // ------------------------------------------------------------

  void
  SolverIf::utilResetCoeffVectors()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    // reset vectors using 'clear' and then 'resize'
    d_ia.clear();
    d_ja.clear();
    d_ar.clear();

    d_ia.resize(s_glpkMargin, 0);            // see notes for constructor
    d_ja.resize(s_glpkMargin, 0);
    d_ar.resize(s_glpkMargin, 0.0);

  } // function 'SolverIf::utilResetCoeffVectors'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : utilResetObjectData (private)
  // ------------------------------------------------------------
  //  Description  : reset the relevant status variables
  //  Role         : utility
  //  Status       : complete
  // ------------------------------------------------------------

  void
  SolverIf::utilResetObjectData()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    // CAUTION: retained: 'd_simplexPresol'

    d_probStatus = status_not_specified;
    d_objSense   = sense_not_specified;
    d_probKlass  = prob_not_specified;
    d_solverType = solver_other;
    d_solnStatus = soln_undefined;
    d_solverRet  = 0;
    d_solnRet    = 0;
```

```
  d_solverCalls = 0;
  d_alertCount  = 0;

} // function 'SolverIf::utilResetObjectData'

// -----------------------------------------------------------
//  MEMBER FUNCTION : utilReportOnProb (private)
// -----------------------------------------------------------
//  Description  : reports on problem
//  Role         : passive utility function -- can be omitted without side-effect
//  Techniques   : read code
//  Status       : complete
//
//  Design notes
//
//      These "utilReportOn.." functions should probably take
//      in an ostream object, rather than return a string.
//
// -----------------------------------------------------------

std::string
SolverIf::utilReportOnProb()
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function", "");

  std::ostringstream ssBuff;
  std::string buf;

  //    EXAMPLE:
  //    problem name                   : prob-some-name
  //    problem klass                  : linear
  //    objective sense                : maximize
  //    problem status                 : solvable
  //    alert count                    :       99      <
  //    solver call count              :        1
  //    strictly binary variables      :        0
  //    strictly integer variables     :        0
  //    strictly continuous variables  :        4
  //    all variables                  :        4
  //    all constraints                :        4
  //    all non-zero coefficients      :       11
  //    coefficients/constraints       :     2.75

  buf = "(faulty code)";
  buf = glp_get_prob_name(d_prob);         // GLPK API
  formatLineOutput(ssBuff, "problem name", buf);

  // previous code used switch (lpx_get_class(d_prob) ..
  // this API is depreciated, but it seems sensible to run an update
  updateProbKlass();
  buf = "(faulty code)";                   // in case this is not overwritten
  switch ( d_probKlass )
    {
    case prob_not_specified:  buf = "not specified";                              break;
    case prob_linear:         buf = "linear";                                     break;
    case prob_mixed_integer:  buf = "mixed integer";                              break;
    case prob_mixed_zero_one: buf = "mixed 0-1";                                  break;
    case prob_pure_integer:   buf = "pure integer";                               break;
    case prob_pure_zero_one:  buf = "pure 0-1";                                   break;
    default:
      std::clog << "** coding error 20 in source file " << __FILE__ << std::endl;
      break;
    }
  formatLineOutput(ssBuff, "problem klass", buf);

  buf = "(faulty code)";                   // in case this is not overwritten
  switch ( glp_get_obj_dir(d_prob) )
    {
    case GLP_MIN: buf = "minimize";                                               break;
    case GLP_MAX: buf = "maximize";                                               break;
    }
  formatLineOutput(ssBuff, "objective sense", buf);

  buf = "(faulty code)";                   // in case this is not overwritten
  switch ( d_probStatus )
    {
    case status_not_specified: buf = "not specified";                             break;
    case status_incomplete:    buf = "incomplete";                                break;
    case status_empty:         buf = "empty (no rows and no cols)";               break;
    case status_solvable:      buf = "solvable";                                  break;
```

```
      case status_submitted:     buf = "submitted";                                  break;
      default:
        std::clog << "** coding error 21 in source file " << __FILE__ << std::endl;
        break;
      }
    formatLineOutput(ssBuff, "problem status", buf);

    std::string extra = "";
    if ( d_alertCount > 0 ) extra = "      <";
    formatLineOutput(ssBuff, "alert count", d_alertCount, extra);

    formatLineOutput(ssBuff, "solver call count", d_solverCalls);

    int nBins  = glp_get_num_bin(d_prob);    // number of integer columns ranged [0,1]
    int nInts  = glp_get_num_int(d_prob);    // number of integer columns

    int nCols  = glp_get_num_cols(d_prob);   // number of columns
    int nRows  = glp_get_num_rows(d_prob);   // number of rows
    int nCoefs = glp_get_num_nz(d_prob);     // number of constraint coefficients
    double div = static_cast<double>(nCoefs)/static_cast<double>(nRows);

    formatLineOutput(ssBuff, "strictly binary variables",     nBins);
    formatLineOutput(ssBuff, "strictly integer variables",    nInts - nBins);
    formatLineOutput(ssBuff, "strictly continuous variables", nCols - nInts);
    formatLineOutput(ssBuff, "all variables",                 nCols);
    formatLineOutput(ssBuff, "all constraints",               nRows);
    formatLineOutput(ssBuff, "all non-zero coefficients",     nCoefs);
    if ( nRows == 0 )
      formatLineOutput(ssBuff, "coefficients/constraints",      "-"); // otherwise 'nan'
    else
      formatLineOutput(ssBuff, "coefficients/constraints",      div);

    return ssBuff.str();

  } // function 'SolverIf::utilReportOnProb'

  // -----------------------------------------------------------
  //  MEMBER FUNCTION : utilReportOnSolver (private)
  // -----------------------------------------------------------
  //  Description  : reports on solver
  //  Role         : passive utility function -- can be omitted without side-effect
  //  Techniques   : read code
  //  Status       : complete
  // -----------------------------------------------------------

  std::string
  SolverIf::utilReportOnSolver()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    std::ostringstream ssBuff;
    std::string buf = "";

    //    EXAMPLE:
    //    solver type used               : mixed integer (with simplex)
    //    simplex presolver              : yes
    //    solver exit                    : some explanation (GLP_ECODE)

    buf = "(faulty code)";                   // in case this is not overwritten
    switch ( d_solverType )
      {
      case solver_simplex:  buf = "simplex";                                       break;
      case solver_interior: buf = "interior point";                                break;
      case solver_integer:  buf = "mixed integer (with simplex)";                  break;
      default:
        break;
      }

    formatLineOutput(ssBuff, "solver type used", buf);

    buf = "(faulty code)";                   // in case this is not overwritten
    switch ( d_simplexPresol )
      {
      case true:  buf = "yes";                                                     break;
      case false: buf = "no";                                                      break;
      default:
        break;
      }
    switch ( d_solverType )
      {
```

```
      case solver_interior:  buf = "not supported as presolver";                    break;
      default:
        break;
      }

    formatLineOutput(ssBuff, "simplex presolver set", buf);

    buf = "(faulty code)";                  // in case this is not overwritten
    switch ( d_solverType )
      {
      case solver_simplex:                  // 'glp_simplex' call documentation
        switch ( d_solverRet )
          {
          case 0:          buf = "solver completed (return 0)";                     break;
          case GLP_EBADB:  buf = "no start as basis invalid (GLP_EBADB)";           break;
          case GLP_ESING:  buf = "no start as basis singular (GLP_ESING)";          break;
          case GLP_ECOND:  buf = "no start as basis ill-conditioned (GLP_ECOND)";   break;
          case GLP_EBOUND: buf = "no start due to bad bounds (GLP_EBOUND)";         break;
          case GLP_EFAIL:  buf = "unspecified solver failure (GLP_EFAIL)";          break;
          case GLP_EOBJLL: buf = "search end as z decreasing (GLP_EOBJLL)";         break;
          case GLP_EOBJUL: buf = "search end as z increasing (GLP_EOBJUL)";         break;
          case GLP_EITLIM: buf = "iteration limit reached (GLP_EITLIM)";            break;
          case GLP_ETMLIM: buf = "time limit reached (GLP_ETMLIM)";                 break;
          case GLP_ENOPFS: buf = "(presolver used) no LP primal (GLP_ENOPFS)";      break;
          case GLP_ENODFS: buf = "(presolver used) no LP dual (GLP_ENODFS";         break;
          default:
            std::clog << "** coding error 22 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      case solver_interior:                 // 'glp_interior' call documentation
        switch ( d_solverRet )
          {
          case 0:           buf = "solver completed (return 0)";                    break;
          case GLP_EFAIL:   buf = "bad data (GLP_EFAIL)";                           break;
          case GLP_ENOCVG:  buf = "slow convergence or divergence (GLP_ENOCVG)";    break;
          case GLP_EITLIM:  buf = "iteration limit exceeded (GLP_EITLIM)";          break;
          case GLP_EINSTAB: buf = "numerical instability (GLP_EINSTAB)";            break;
          default:
            std::clog << "** coding error 23 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      case solver_integer:                  // 'glp_intopt' call documentation
        switch ( d_solverRet )
          {
          case 0:          buf = "solver completed (returned 0)";                   break;
          case GLP_EBOUND: buf = "no start due to bad bounds (GLP_EBOUND)";         break;
          case GLP_EROOT:  buf = "no start as no initial LP relax (GLP_EROOT)";     break;
          case GLP_EFAIL:  buf = "unspecified solver failure (GLP_EFAIL)";          break;
          case GLP_ETMLIM: buf = "time limit reached (GLP_ETMLIM)";                 break;
          case GLP_ESTOP:  buf = "search terminated by application (GLP_ESTOP)";    break;
          default:
            std::clog << "** coding error 24 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      default:
        break;
      }

    formatLineOutput(ssBuff, "exit status", buf);

    if ( d_solverType == solver_simplex )    // possibly too restrictive
      {
        int niter = lpx_get_int_parm(d_prob, LPX_K_ITCNT);  // lpx is correct
        formatLineOutput(ssBuff, "simplex iterations", niter);
      }

    return ssBuff.str();

  } // function 'SolverIf::utilReportOnSolver'

  // -------------------------------------------------------
  //  MEMBER FUNCTION : utilReportOnSoln (private)
  // -------------------------------------------------------
  //  Description  : reports on solution
  //  Role         : passive utility function -- can be omitted without side-effect
  //  Techniques   :
  //  Status       : complete
  // -------------------------------------------------------
```

```
  std::string
  SolverIf::utilReportOnSoln()
  {
    // initial reporting
    s_logger->repx(logga::adhc, "entering member function", "");

    std::ostringstream ssBuff;
    std::string buf = "";

    //    EXAMPLE:
    //    solution status                : optimal solution
    //    objective                      :     2.99e+002

    buf = "(faulty code)";                   // in case this is not overwritten
    switch ( d_solverType )
      {
      case solver_simplex:                   // 'glp_get_status' call documentation
        switch ( d_solnRet )
          {
          case GLP_OPT:    buf = "optimal soln (GLP_OPT)";                          break;
          case GLP_FEAS:   buf = "feasible soln (GLP_FEAS)";                        break;
          case GLP_INFEAS: buf = "soln infeasible but not proven bad (GLP_INFEAS)"; break;
          case GLP_NOFEAS: buf = "problem proven nonfeasible (GLP_NOFEAS)";         break;
          case GLP_UNBND:  buf = "problem unbounded (GLP_UNBND)";                   break;
          case GLP_UNDEF:  buf = "soln undefined (GLP_UNDEF)";                      break;
          default:
            std::clog << "** coding error 25 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      case solver_interior:                  // 'glp_ipt_status' call documentation
        switch ( d_solnRet )
          {
          case GLP_OPT:   buf = "optimal soln (GLP_OPT)";                           break;
          case GLP_UNDEF: buf = "soln undefined (GLP_UNDEF)";                       break;
            // two new returns added in GLPK 4.38
#if ( GLP_MAJOR_VERSION == 4 && GLP_MINOR_VERSION >= 38 )
          case GLP_INFEAS: buf = "soln infeasible but not proven bad (GLP_INFEAS)"; break;
          case GLP_NOFEAS: buf = "problem proven nonfeasible (GLP_NOFEAS)";         break;
#endif
          default:
            std::clog << "** coding error 26 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      case solver_integer:                   // 'glp_mip_status' call documentation
        switch ( d_solnRet )
          {
          case GLP_OPT:    buf = "integer optimal soln (GLP_OPT)";                  break;
          case GLP_FEAS:   buf = "integer feasible soln, maybe optimal (GLP_FEAS)"; break;
          case GLP_NOFEAS: buf = "problem proven infeasible (GLP_NOFEAS)";          break;
          case GLP_UNDEF:  buf = "soln undefined (GLP_UNDEF)";                      break;
          default:
            std::clog << "** coding error 27 in source file " << __FILE__ << std::endl;
            break;
          }
        break;
      default:
        break;
      }
    formatLineOutput(ssBuff, "solution status", buf);

    double objective = getObjectiveValue();
    formatLineOutput(ssBuff, "final objective value", objective);

    return ssBuff.str();

  } // function 'SolverIf::utilReportOnSoln'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : indexToString <> (private)
  // ------------------------------------------------------------
  //  Description  : convert a number to a string
  //  Role         : utility function
  //  Techniques   : ostream formatting
  //  Status       : complete
  //
  //  Design notes
  //
  //      Should non-default formatting be required, Stephens
```

```
//      etal (2006) cover the formatting of floating point
//      output in recipe 10.2 (pp 356--359).
//
//      Alternatively, implement (for yourself) code using the
//      Boost library Boost.Conversion and
//      boost::lexical_cast<std::string>(number).
//
// -------------------------------------------------------

template<typename T>
std::string
SolverIf::numToString
(const T number) const
{
  std::ostringstream ssBuf;
  ssBuf << number;                          // specify additional formatting here
  return ssBuf.str();
}

// -------------------------------------------------------
//  MEMBER FUNCTION : indexToString (private)
// -------------------------------------------------------
//  Description  : index to string conversion
//  Role         : reporting support
//  Techniques   : C++ streams
//  Status       : complete
//
//  Design notes
//
//       The int 's_indexPad' is set in the class definition.
//       A value of 0 (or 1) suppress padding, larger values
//       enable zero padding.
//
//  CAUTION: negative indexes
//
//      The padding routine requires non-negative input
//      (otherwise it can output strings like: 00-22).  The
//      Boost.Format library provides a more robust solution
//      (or a more sophisticated routine can be hand coded).
//
// -------------------------------------------------------

std::string
SolverIf::indexToString
(const int index) const
{
  std::ostringstream ssBuf;
  ssBuf << std::setfill('0') << std::setw(s_indexPad) << index;
  return ssBuf.str();
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getProblemTag
// -------------------------------------------------------
//  Description  : return problem tag as submitted on construction
//  Role         : general information
//  Status       : complete
//  CAUTION      : consider also 'getProblemLabel'
// -------------------------------------------------------

const std::string
SolverIf::getProblemTag() const
{
  return d_probTag;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : getAlertCount
// -------------------------------------------------------
//  Description  : return number of alerts
//  Role         : general information
//  Status       : complete
// -------------------------------------------------------

int
SolverIf::getAlertCount() const
{
  return d_alertCount;
}

// -------------------------------------------------------
```

```
//  MEMBER FUNCTION : incrementAlertCount (private)
// ---------------------------------------------------------
//  Description  : increments alert count and offers additional features
//  Role         : use in preference to 'd_alertCount++'
//  Techniques   : utility function
//  Status       : complete
// ---------------------------------------------------------

int
SolverIf::incrementAlertCount()
{
  d_alertCount++;                           // this should be a nested class ideally
  if ( ! svif::DBUG )
    {
      s_logger->repx(logga::info, "incrementing alerts, alert count", d_alertCount);
    }

  return d_alertCount;
}

// ---------------------------------------------------------
//  MEMBER FUNCTION : formatLineOutput (private)
//  MEMBER FUNCTION : formatLineOutput <> (private)
// ---------------------------------------------------------
//  Description  : provide formatted output for use in 'utilReportOn*' functions
//  Role         : utility function
//  Techniques   : string-stream
//  Status       : complete
// ---------------------------------------------------------

void
SolverIf::formatLineOutput
(std::ostringstream& ssOut,
 std::string         tag,
 std::string         value)
{
  std::ios_base::fmtflags was = ssOut.flags();
  ssOut << std::setw(s_W0)
        << ""
        << std::setw(s_W1)
        << std::left
        << tag
        << std::setw(s_W2)
        << ":"
        << value
        << "\n";
  ssOut.flags(was);                         // reset formatting flags
}

template <typename T>
void
SolverIf::formatLineOutput
(std::ostringstream& ssOut,
 std::string         tag,
 T                   value,                // int or double expected
 std::string         extra)                // optional trailing string
{
  std::ios_base::fmtflags was = ssOut.flags();
  ssOut << std::setprecision(s_P1);
  ssOut << std::setw(s_W0)
        << ""
        << std::setw(s_W1)
        << std::left
        << tag
        << std::setw(s_W2)
        << ":"
        << std::setw(s_W3)
        << std::right
        << value
        << extra
        << "\n";
  ssOut.flags(was);                         // reset formatting flags
}

// ---------------------------------------------------------
//  MEMBER FUNCTION : getGlpkVersion (static)
// ---------------------------------------------------------
//  Description  : returns GLPK version string in form 0.00
//  Role         : general information
//  Techniques   : 'glp_version'
//  Status       : complete
```

```
// ----------------------------------------------------------

const std::string
SolverIf::getGlpkVersion()
{
  return glp_version();                    // const char* converted to std::string
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : updateProbKlass (private)
// ----------------------------------------------------------
//  Description  : update the problem klass based on the current build information
//  Role         : internal
//  Techniques   : 'd_probKlass'
//  Status       : complete
//
//  CAUTION: terminology
//
//      The term 'integer' here excludes binaries -- and thus
//      differs from from the problem classification!
//
// ----------------------------------------------------------

void
SolverIf::updateProbKlass()
{
  // code
  int nVars     = 0;                       // variables
  int nCons     = 0;                       // continuous variables
  int nNonCons  = 0;                       // integer and binary variables
  int nInts     = 0;                       // integer (excluding binary) variables
  int nBins     = 0;                       // binary variables

  nVars    = glp_get_num_cols(d_prob);     // continuous, integer, and binary variables
  nNonCons = glp_get_num_int(d_prob);      // integer and binary variables
  nBins    = glp_get_num_bin(d_prob);      // just binary variables

  nCons = nVars    - nNonCons;
  nInts = nNonCons - nBins;

  ProblemKlass klass = prob_not_specified; // initial value

  if ( nVars == 0 )                        // no variables loaded
    {
      klass = prob_linear;                 // an empty problem is deemed linear!
    }
  else                                     // a substantial problem
    {
      if ( nCons == 0 )                    // pure not mixed nonlinear
        {
          if ( nInts == 0 )                // no integers present
            klass = prob_pure_zero_one;
          else                             // integers present
            klass = prob_pure_integer;
        }
      else if ( nNonCons == 0 )            // linear
        {
          klass = prob_linear;
        }
      else                                 // mixed nonlinear
        {
          if ( nInts == 0 )                // no integers present
            klass = prob_mixed_zero_one;
          else                             // integers present
            klass = prob_mixed_integer;
        }
    }

  // report
  std::string buf = "(faulty code)";       // in case this is not overwritten
  switch ( klass )
    {
    case prob_not_specified:  buf = "not specified";                              break;
    case prob_linear:         buf = "linear";                                     break;
    case prob_mixed_integer:  buf = "mixed integer";                              break;
    case prob_mixed_zero_one: buf = "mixed 0-1";                                  break;
    case prob_pure_integer:   buf = "pure integer";                               break;
    case prob_pure_zero_one:  buf = "pure 0-1";                                   break;
    default:
      std::clog << "** coding error 28 in source file " << __FILE__ << std::endl;
      break;
```

```
      }
    s_logger->repx(logga::xtra, "klass = " + buf, "");

    // return
    d_probKlass = klass;

  } // function 'SolverIf::updateProbKlass'

  // ------------------------------------------------------------
  //  MEMBER FUNCTION : updateProbStatus (private)
  // ------------------------------------------------------------
  //  Description  : update the problem status based on the current build information
  //  Role         : internal
  //  Techniques   : 'd_probStatus'
  //  Status       : complete
  // ------------------------------------------------------------

  void
  SolverIf::updateProbStatus()
  {
    // initial reporting omitted as not helpful

    int nVars   = glp_get_num_rows(d_prob);  // all variables
    int nCons   = glp_get_num_cols(d_prob);  // all constraints
    int nCoeffs = glp_get_num_nz(d_prob);    // loaded coefficients

    ProblemStatus status = status_not_specified;  // initial value

    // process (see commit r3718 for version which rejects empty problems)
    if      ( nVars      > 0 &&
              nCons      > 0 &&
              nCoeffs    > 0 &&
              d_objSense > sense_not_specified )  // overwrite is by 'setConstraintSense'
      {
        status = status_solvable;
        s_logger->repx(logga::xtra, "status = solvable", "");
      }
    else if ( nVars      == 0 &&
              nCons      == 0 &&
              nCoeffs    == 0 &&
              d_objSense > sense_not_specified )  // overwrite is by 'setConstraintSense'
      {
        status = status_empty;
        s_logger->repx(logga::xtra, "status = empty", "");
      }
    else
      {
        status = status_incomplete;
        s_logger->repx(logga::adhc, "status = incomplete", "");
      }

    d_probStatus = status;                   // update

  } // function 'SolverIf::updateProbStatus'

} // namespace svif

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : context.cc
//  file-create-date : Wed 06-May-2009 21:11 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : abstract context entity / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/context.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "context.h"          // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : Context
// ----------------------------------------------------------

Context::Context
(const std::string entityId,
 Record&           record) :
  FullEntity(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

Context::~Context()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  CLASS           : AmbientConditions
// ----------------------------------------------------------

AmbientConditions::AmbientConditions
(const std::string entityId,
 Record&           record) :
  Context(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}
```

```
AmbientConditions::~AmbientConditions()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  CLASS           : EconomicContext
// ----------------------------------------------------------

EconomicContext::EconomicContext
(const std::string entityId,
 Record&           record) :
  Context(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

EconomicContext::~EconomicContext()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  CLASS           : PublicPolicyContext
// ----------------------------------------------------------

PublicPolicyContext::PublicPolicyContext
(const std::string entityId,
 Record&           record) :
  Context(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

PublicPolicyContext::~PublicPolicyContext()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxamb01.cc
//  file-create-date : Thu 07-May-2009 08:15 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete ambient conditions contexts 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxamb01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "cxamb01.h"          // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <algorithm>          // STL copying, searching, and sorting
#include <limits>             // numeric_limits<T>::infinity() and similar
#include <numeric>            // STL numerical algorithms
#include <sstream>            // string-streams
#include <string>             // C++ strings

#include <cmath>              // C-style maths, ceil(), floor(), sqrt()

#include <boost/format.hpp>                       // printf style formatting
#include <boost/math/constants/constants.hpp>     // high-precision mathematical constants

//  CODE

// ----------------------------------------------------------
//  CLASS (FUNCTOR) : ::SynWindMaker
// ----------------------------------------------------------
//  Description  : makes synthetic wind speed data
//  Role         : support for ambient air contexts based on internal simulation
//  Techniques   : functor, Boost.Math library, Boost.Random library
//  Status       : complete (but see caveats, particularly regarding volatility)
//
//  Design notes
//
//      Parameters:
//
//          mean  : mean wind speed (by convention in [m/s] at 10m mast height)
//
//      Output:
//
//          random wind speed which conforms to a Rayleigh distribution
//
//          Whether the output is repeatable depends on whether
```

```
//          or not the '--pepper' option is set.  In other words,
//          the random engine either relies on its default
//          internal state or is explicitly seeded on the FIRST
//          constructor call.
//
//      Typical usage:
//
//          std::vector<double> winds(10);      // timeseries of ten intervals
//          double averageWindSpeed = 10.5;     // see above for the underlying protocol
//          std::generate(winds.begin(),        // see <algorithm>
//                        winds.end(),
//                        ::SynWindMaker(averageWindSpeed));
//
//          Note carefully that the 'SynWindMaker' functor is
//          instantiated only once for the above task -- and is
//          not created anew for every element processed.
//
//  Caveats related to this implementation
//
//      Statistical issues
//
//          This method assumes independent sampling -- which is
//          readily visible in the fractured results and is
//          clearly not a good representation for real wind data
//          which is substantially attenuated by the prior wind
//          state.
//
//          That said, this code does represent a reasonable
//          first pass at synthetic wind and should also form a
//          reasonable basis for future development.
//
//          Indeed, this functor will give the correct annual
//          energy output, considering the highly nonlinear power
//          characteristics of wind turbines.
//
//          Note too that the 'Weibull' distribution, one of the
//          generalizations of the 'Rayleigh' distribution (as
//          used here), can better match real wind data -- but
//          also requires an additional characterizing parameter.
//
//      Software design
//
//          Random numbers conforming to a particular
//          distribution should not, as a rule, be generated by
//          "uniform sampling".  In fact, the Boost.Math
//          documentation states:
//
//             "Random numbers that approximate Quantiles of
//              Distributions
//
//              If you want random numbers that are distributed
//              in a specific way, for example in a uniform,
//              normal or triangular, see Boost.Random.
//
//              Whilst in principal there's nothing to prevent
//              you from using the quantile function to convert a
//              uniformly distributed random number to another
//              distribution, in practice there are much more
//              efficient algorithms available that are specific
//              to random number generation."
//
//          At the time of writing, Boost.Random did not support
//          the 'Rayleigh' distribution.
//
//  Further information
//
//      Wikipedia on wind energy and related topics.
//
//      Becker (2007 pp336-342) for use of the
//      'boost::variate_generator' class template specifically
//      and the Boost.Random library more generally.
//
//      Stephens etal (2006 pp407-410) on aspects of the software
//      design used here.
//
//      The Boost libraries documentation including Boost.Math >
//      Statistical Distributions Reference > Non-Member
//      Properties.
//
//  CAUTION: copy constructor
//
```

```
//     An accessible and working copy constructor is required.
//
// ----------------------------------------------------------

namespace xeona
{
  // CREATORS

  SynWindMaker::SynWindMaker
  (const double mean) :
    d_callCount(0),                          // initialize to zero
    d_mean(mean),
    d_raydis(d_mean * sqrt(2.0 / boost::math::constants::pi<double>()))    // [1]
  {
    s_logger->repx(logga::dbug, "constructor call, d_mean ", d_mean);
    s_logger->repx(logga::dbug, "Rayleigh distribution sigma", d_raydis.sigma());

    // first call seeding as required
    if ( s_firstCall )
      {
        if ( xeona::pepper == true )             // controls which seed to use
          {
            std::time_t stamp = std::time(0);     // CAUTION: std::time is not ideal
            boost::mt19937::result_type seed      // better than hardcoding 'unsigned'
              = static_cast<boost::mt19937::result_type>(stamp);
            s_rng.engine().seed(seed);            // CAUTION: note the form of this call
            s_firstCall = false;
            s_logger->repx(logga::dbug, "first call and seeding with", seed);
          }
        else
          {
            s_firstCall = false;
            s_logger->repx(logga::xtra, "first call but using default seed", "");
          }
      }
  }

  // [1] for the Rayleigh distribution:
  //                        ____
  //                       / pi
  //      mean = sigma    / ---
  //                   \/   2
  //
  // but here we want 'sigma', the sole distribution parameter

  SynWindMaker::~SynWindMaker()
  {
    s_logger->repx(logga::adhc, "destructor call, functor call count", d_callCount);
  }

  // MANIPULATORS

  double                                     // wind value in [m/s]
  SynWindMaker::operator()() const
    throw(std::domain_error, std::overflow_error)    // exception specification
  {
    ++d_callCount;

    // ---------------------------------
    //  generate uniform random number
    // ---------------------------------

    // generate uniform random number on interval [0,1)
    //
    // the following call does this, which is useful because
    // prob = 1.0 is not admissible for the Rayleigh
    // distribution, that is, no finite value exists for an
    // inverse cumulative probability of unity

    const double randomUniformProb = s_rng();

    // ---------------------------------
    //  generate Rayleigh-compliant data
    // ---------------------------------

    try
      {
        return boost::math::quantile(d_raydis, randomUniformProb);
      }
    catch( const std::domain_error& e )
      {
```

```
        // a probability outside [0,1] supplied (should never
        // be here as such values cannot be generated)
        s_logger->repx(logga::warn, "given probability not [0,1]", randomUniformProb);
        throw;                          // rethrow, CAUTION: usage is correct
      }
    catch( const std::overflow_error& e )
      {
        // no finite value for the specified probability
        // (should never be here as unity cannot be generated)
        s_logger->repx(logga::warn, "invalid Rayleigh probability", randomUniformProb);
        throw;                          // rethrow
      }
  }

  // STATIC DEFINITIONS

  bool SynWindMaker::s_firstCall = true;
  boost::mt19937 SynWindMaker::s_engine;
  boost::uniform_real<double> SynWindMaker::s_unidis;
  boost::variate_generator<boost::mt19937, boost::uniform_real<double> >
  SynWindMaker::s_rng(SynWindMaker::s_engine, SynWindMaker::s_unidis);

  logga::spLogger SynWindMaker::s_logger = logga::ptrLogStream();

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS           : CxAmbientAir (abstract)
// ----------------------------------------------------------

CxAmbientAir::CxAmbientAir
(const std::string entityId,
 Record&           record) :
  AmbientConditions(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CxAmbientAir::~CxAmbientAir()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  CLASS           : CxAmbientAirTs
// ----------------------------------------------------------

CxAmbientAirTs::CxAmbientAirTs
(const std::string entityId,
 Record&           record) :
  CxAmbientAir(entityId, record),
  d_windSpeeds(record.tieTimeseries<double>("wind-speeds")),
  d_airTemps(record.tieTimeseries<double>("air-temps"))
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

CxAmbientAirTs::~CxAmbientAirTs()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

double
CxAmbientAirTs::getWindSpeed
(const int step) const
{
  return d_windSpeeds->at(step);
}

double
CxAmbientAirTs::getWindSpeedPeek
(const int step,
 const int ahead) const
{
  try
    {
      return d_windSpeeds->at(step + ahead);
    }
  catch( const std::out_of_range& e )        // approaching the timeseries tail
    {
```

```
      try
        {
          return d_windSpeeds->at(ahead - 1);     // roll around (minus one for zero-base)
        }
      catch( const std::out_of_range& e )    // still too little information
        {
          std::ostringstream oss;
          oss << step << " " << ahead << " " << d_windSpeeds->size();
          s_logger->repx(logga::warn, "out of range, step ahead size", oss.str());
          return std::numeric_limits<double>::quiet_NaN();
        }
    }
}

double
CxAmbientAirTs::getAirTemp
(const int step) const
{
  return d_airTemps->at(step);
}

double
CxAmbientAirTs::getAirTempPeek
(const int step,
 const int ahead) const
{
  try
    {
      return d_airTemps->at(step + ahead);
    }
  catch( const std::out_of_range& e )        // approaching the timeseries tail
    {
      try
        {
          return d_airTemps->at(ahead - 1);  // roll around (minus one for zero-base)
        }
      catch( const std::out_of_range& e )    // still too little information
        {
          std::ostringstream oss;
          oss << step << " " << ahead << " " << d_airTemps->size();
          s_logger->repx(logga::warn, "out of range, step ahead size", oss.str());
          return std::numeric_limits<double>::quiet_NaN();
        }
    }
}

// ------------------------------------------------------------
//  CLASS           : CxAmbientAirSim
// ------------------------------------------------------------

CxAmbientAirSim::CxAmbientAirSim
(const std::string entityId,
 Record&           record) :
  CxAmbientAir(entityId, record),
  d_meanWindSpeed(record.tieSingle<double>("mean-wind-speed")),
  d_constantAirTemp(record.tieSingle<double>("constant-air-temp")),
  d_windSpeeds(record.tieTimeseries<double>("wind-speeds"))
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";

  // generate synthetic timeseries
  std::generate(d_windSpeeds->begin(),
                d_windSpeeds->end(),
                xeona::SynWindMaker(d_meanWindSpeed));

  // check mean
  const int    length = d_windSpeeds->size();
  const double sum    = std::accumulate(d_windSpeeds->begin(),   // refer <numeric>
                                        d_windSpeeds->end(),
                                        0.0);     // initial value
  // report
  std::ostringstream oss;
  oss << length << "  " << d_meanWindSpeed << "  " << (sum / length) << "\n";
  s_logger->repx(logga::xtra, "size, read-in mean, calculated mean", oss.str());
}

CxAmbientAirSim::~CxAmbientAirSim()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}
```

```
double
CxAmbientAirSim::getWindSpeed
(const int step) const
{
  const double windSpeed = d_windSpeeds->at(step);
  std::ostringstream oss;
  oss << windSpeed << " (" << step << ")";
  s_logger->repx(logga::adhc, "simulated wind-speed (step)", oss.str());
  return windSpeed;
}

double
CxAmbientAirSim::getWindSpeedPeek
(const int step,
 const int ahead) const
{
  try
    {
      return d_windSpeeds->at(step + ahead);
    }
  catch( const std::out_of_range& e )        // approaching the timeseries tail
    {
      try
        {
          return d_windSpeeds->at(ahead - 1);     // roll around (minus one for zero-base)
        }
      catch( const std::out_of_range& e )    // still too little information
        {
          std::ostringstream oss;
          oss << step << " " << ahead << " " << d_windSpeeds->size();
          s_logger->repx(logga::warn, "out of range, step ahead size", oss.str());
          return std::numeric_limits<double>::quiet_NaN();
        }
    }
}

double
CxAmbientAirSim::getAirTemp
(const int step) const
{
  return d_constantAirTemp;
}

double
CxAmbientAirSim::getAirTempPeek
(const int step,
 const int ahead) const
{
  return d_constantAirTemp;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxamb02.cc
//  file-create-date : Thu 07-May-2009 13:52 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete ambient conditions contexts 2 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxamb02.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "cxamb02.h"          // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <limits>             // numeric_limits<T>::infinity() and similar
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : CxAmbientSolar (abstract)
// ----------------------------------------------------------

CxAmbientSolar::CxAmbientSolar
(const std::string entityId,
 Record&           record) :
  AmbientConditions(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CxAmbientSolar::~CxAmbientSolar()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  CLASS           : CxAmbientSolarTs
// ----------------------------------------------------------

CxAmbientSolarTs::CxAmbientSolarTs
(const std::string entityId,
 Record&           record) :
  CxAmbientSolar(entityId, record),
```

```
  d_solarDirect(record.tieTimeseries<double>("solar-direct")),
  d_solarDifuse(record.tieTimeseries<double>("solar-diffuse"))
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
}

CxAmbientSolarTs::~CxAmbientSolarTs()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

boost::tuple
<double,                                    // direct component
 double>                                    // diffuse component
CxAmbientSolarTs::getSolar
(const int step) const
{
  return boost::make_tuple
    (d_solarDirect->at(step),
     d_solarDifuse->at(step));
}

boost::tuple
<double,                                    // direct component
 double>                                    // diffuse component
CxAmbientSolarTs::getSolarPeek
(const int step,
 const int ahead) const
{
  try
    {
      return boost::make_tuple
        (d_solarDirect->at(step + ahead),
         d_solarDifuse->at(step + ahead));
    }
  catch( const std::out_of_range& e )       // approaching the timeseries tail
    {
      try
        {
          return boost::make_tuple
            (d_solarDirect->at(ahead - 1),  // roll around (minus one for zero-base)
             d_solarDifuse->at(ahead - 1));
        }
      catch( const std::out_of_range& e )   // still too little information
        {
          std::ostringstream oss;
          oss << step << " " << ahead << " " << d_solarDirect->size();
          s_logger->repx(logga::warn, "out of range, step ahead size", oss.str());
          return boost::make_tuple
            (std::numeric_limits<double>::quiet_NaN(),
             std::numeric_limits<double>::quiet_NaN());
        }
    }
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxecon01.cc
//  file-create-date : Thu 07-May-2009 08:16 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete economic contexts 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxecon01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "cxecon01.h"         // companion header for this file (place first)

#include "../c/recset.h"      // records and fields and also record-sets

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : CxCommercial (abstract)
// ----------------------------------------------------------

CxCommercial::CxCommercial
(const std::string entityId,
 Record&           record) :
  EconomicContext(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CxCommercial::~CxCommercial()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

// ----------------------------------------------------------
//  CLASS           : CxCommercialFix
// ----------------------------------------------------------

CxCommercialFix::CxCommercialFix
(const std::string entityId,
 Record&           record) :
  CxCommercial(entityId, record),
  d_comInterestRate(record.tieSingle<double>("commercial-interest-rate"))
```

```
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CxCommercialFix::~CxCommercialFix()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

double                                  // decimal form unit [-/y], say 0.2
CxCommercialFix::getComInterestRate() const
{
  return d_comInterestRate;
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cxpubpol01.cc
//  file-create-date : Thu 07-May-2009 08:27 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : concrete public policy contexts 1 / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/e/cxpol01.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "cxpol01.h"          // companion header for this file (place first)

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

// ----------------------------------------------------------
//  CLASS           : CxPublicPolicyA
// ----------------------------------------------------------

CxPublicPolicyA::CxPublicPolicyA
(const std::string entityId,
 Record&           record) :
  PublicPolicyContext(entityId, record)
{
  s_logger->repx(logga::xtra, "constructor call", getIdAndKind());
  d_builtinRemark = "beta";
}

CxPublicPolicyA::~CxPublicPolicyA()
{
  s_logger->repx(logga::xtra, "destructor call", getIdAndKind());
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : cta.cc
//  file-create-date : Fri 06-Feb-2009 15:12 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : captrans algorithm / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/cta.cc $

//  LOCAL AND SYSTEM INCLUDES

#include "cta.h"              // companion header for this file (place first)

#include "../c/util2.h"       // free functions which offer general utilities 2
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../b/gate.h"        // gateway entity
#include "../b/domcon.h"      // domain controller entity
#include "../a/recorder.h"    // summarizing recorder class

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// ------------------------------------------------------------
//  FREE FUNCTION   : ::formatLeader
// ------------------------------------------------------------
//  Description  : returns formatted leader string like "  CAPSET-01 (00)"
//  Role         : support reporting
//  Techniques   : string stream
//  Status       : complete
// ------------------------------------------------------------

namespace
{
  std::string
  formatLead
  (
   const std::string& label,
   const int          step,
   const int          count)
  {
    const int pad = 2;
    std::ostringstream oss;
    oss << "  "
        << label
        << "-"
```

```
        << std::setw(pad) << std::setfill('0') << count
        << " ("
        << std::setw(pad) << std::setfill('0') << step
        << ")";
    return oss.str();
  }

} // unnamed namespace

// ----------------------------------------------------------
//  CLASS           : CapTransAlg
// ----------------------------------------------------------

// STATIC DEFINITIONS - with and without explicit initialization

logga::spLogger CapTransAlg::s_logger = logga::ptrLogStream();

// MEMBER FUNCTIONS

// ----------------------------------------------------------
//  MEMBER FUNCTION : CapTransAlg
// ----------------------------------------------------------

CapTransAlg::CapTransAlg
(const int                                          step,
 const std::vector<shared_ptr<DomainController> >& domcons) :  // order significant
   d_step(step),
   d_originatingDomcons(domcons)
{
  s_logger->repx(logga::xtra,
                 "constructor call, orig domcon count",
                 d_originatingDomcons.size());
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CapTransAlg
// ----------------------------------------------------------

CapTransAlg::~CapTransAlg()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtaFixed
// ----------------------------------------------------------

CtaFixed::CtaFixed
(const int                                          step,
 const std::vector<shared_ptr<DomainController> >& domcons) :
  CapTransAlg(step, domcons)
{
  s_logger->repx(logga::warn, "not fully implemented", "");
  std::ostringstream put;
  put << "  the \"fixed\" version of the captrans algorithm is not implemented"   << "\n"
      << "  try the \"simple\" or \"hop-relit\" versions instead"                  << "\n";
  s_logger->putx(logga::xtra, put);
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtaFixed::captrans
// ----------------------------------------------------------

tribool                           // 'indeterminate' if no originating domains
CtaFixed::captrans
(const xeona::DomainMode capacityMode)
{
  const tribool ret = false;
  s_logger->repx(logga::warn, "not yet implemented, return", ret);
  return ret;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : CtaSimple
// ----------------------------------------------------------

CtaSimple::CtaSimple
(const int                                          step,
 const std::vector<shared_ptr<DomainController> >& domcons) :
  CapTransAlg(step, domcons)
{
```

```
  s_logger->repx(logga::dbug, "constructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~CtaSimple
// ----------------------------------------------------------

CtaSimple::~CtaSimple()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : captrans
// ----------------------------------------------------------
//  Description  : traverse domain graph and make 'capset' and 'transolve' calls
//  Role         : system solution, called by 'Overseer' singleton
//  Techniques   : graph traversal
//  Status       : complete but not fully tested (also macro-controlled hop-relit code)
//
//  Design notes
//
//      This function implements the "CapTrans" algorithm.
//
//      The commitment domain graph is implicit -- meaning that a
//      dedicated graph container is not used.
//
//  Briefly
//
//        capset     =  gateway capacity setting method
//        transolve  =  domain transaction solving method
//
//      The notion of under-capacitated means that the gateway in
//      question has not yet been capset, but which does of
//      course still possess innate limits as part of its
//      specification.
//
//  Logging and event recording optional
//
//      If need be, member function calls based on logger
//      instance 's_logger' and class 'EventRecorder' instance
//      'ctalog' can be deleted to make the code cleaner.
//
//  CAUTION: hop-relit code here a temporary measure
//
//      The 'XE_CTA' macro is a temporary measure being used
//      during algorithm development.  Later on, the code should
//      be split into two and added to each of the correct
//      classes.  That will thereby allow XEM-level control.
//
// ----------------------------------------------------------

// set the form of algorithm to be used during development
//
//   1 = without relitigation
//   2 = with hop-relitigation

#if !defined(XE_CTA)                          // allows external override via 'XE_CTA'
# define XE_CTA 1                             // set the default, see above for definitions
#endif

// protect against inappropriate macro values

#if   (XE_CTA == 1)
#elif (XE_CTA == 2)
#else
# error "XE_CTA macro not set to a supported value {1,2}"
#endif

tribool                                       // 'indeterminate' if no originating domains
CtaSimple::captrans
(const xeona::DomainMode capacityMode)
{
  // ---------------------------------
  //  preamble
  // ---------------------------------

  int count = 0;                              // output counter
  std::ostringstream put;                     // for logging

  // ---------------------------------
```

```
  // event recorder (for reporting)
  // ---------------------------------

  EventRecorder ctalog("in cta", d_step);    // see 'a/recorder.ut0.cc' for sample usage

  // ---------------------------------
  //  data integrity checks
  // ---------------------------------

  if ( ! xeona::isTwoContained(capacityMode, xeona::e_capacityModes) )
    {
      std::ostringstream oss;
      oss << capacityMode << " " << xeona::e_capacityModes;
      s_logger->repx(logga::warn, "inappropriate capacity mode", oss.str());
      return false;
    }

  if ( d_originatingDomcons.empty() )
    {
      s_logger->repx(logga::rankNoData, "no originating domain controllers", "");
      return indeterminate;
    }

  // ---------------------------------
  //  string format 'step'
  // ---------------------------------

  // for convenience
  const int step = d_step;

  // note the the 'Boost.Format' library does not
  // support the ANSI C 'printf' * width and
  // precision specifier

  const unsigned span = Entity::getHorizonSteps();
  std::string formatString;                       // for 'boost::format' call
  if      ( span <   100 ) formatString = "%02d"; //     02 thru 99
  else if ( span < 10000 ) formatString = "%04d"; //   0002 thru 9999
  else                     formatString = "%06d"; // 000002 and above
  const std::string strStep = boost::str(boost::format(formatString) % step);

  // ---------------------------------
  //  preamble
  // ---------------------------------

  // load the domain controllers vector into a more convenient popable container
  SmartPtrPopper<DomainController> origDomcons;   // simple popable container container
  origDomcons.load(d_originatingDomcons);         // shared pointers elements copied over

  // declare some variables -- these shared pointers are naturally empty
  shared_ptr<DomainController> dc;
  shared_ptr<Gateway>          gw;

  // ---------------------------------
  // variation specific code
  // ---------------------------------

#if   (XE_CTA == 1) // without relitigation
#elif (XE_CTA == 2) // with hop-relitigation

  bool looseflag = false;

#endif // macro 'XE_CTA'

  // ---------------------------------
  //  CTA reset loop
  // ---------------------------------

  // There are two options here:
  //
  //   * deploy a domain graph traversal as seen in
  //     'Overseer::run'
  //
  //   * use looped calls as per 'xeona::registerGates' and its
  //     helper function
  //
  // The second method is currently used here -- but that
  // decision could be revisited.

  std::vector<shared_ptr<DomainController> > domcons = Entity::retDomains();
  BOOST_FOREACH( shared_ptr<DomainController> domcon, domcons )
```

```
    {
      std::vector<shared_ptr<Gateway> > gates = domcon->retSelgates();
      BOOST_FOREACH( shared_ptr<Gateway> gate, gates )
        {
          gate->reset();
        }
    }

  // ---------------------------------
  //  abandon if yeek 29
  // ---------------------------------

  // additional action as appropriate
  // YEEK 29 CODE (set by '--yeek')
  if ( xeona::yeek == 29 )
    {
      std::string msg;
      msg += "part way thru function '";
      msg += __func__;
      msg += "', main loop code next";
      s_logger->repx(logga::xtra, "will throw xeona::yeek_abandon", xeona::yeek);
      throw xeona::yeek_abandon(msg);
    }

  // ---------------------------------
  //  CTA starting position
  // ---------------------------------

  // grab first originating domain
  dc = origDomcons.pop();
  ctalog.event("pop",
               "originating domcon " + dc->getIdAndKind(),
               "cta starting position");

  // ---------------------------------
  //  CTA main loop
  // ---------------------------------

  // CAUTION: the single "=" (copy assignment operator) in the
  // various selection statements ('if' and perhaps 'switch' ) is
  // normally correct

  while ( true )                              // a clear idiom but not the most compact
    {
      s_logger->repx(logga::adhc, "- loop start with domcon", dc->getIdAndKind());
      ctalog.event("loop", "entering main loop", "");

      // FIRST: try to back-track along the next (theta) unmarked buygate

      if ( gw = dc->lowestNoThetaBuy() )          // lowest not-theta'ed buygate
        {
          s_logger->repx(logga::adhc, "1 lowest no theta buy", gw->getIdAndKind());
          gw->markTheta();                        // mark theta
          s_logger->repx(logga::adhc, "1 buygate now theta'ed", gw->getTheta());
          shared_ptr<DomainController> from = dc; // used by 'ctalog'
          ctalog.mark("theta", gw, dc, "lowest no theta buy now theta'ed");
          dc = gw->hop(dc);                       // sel-side hop
          s_logger->repx(logga::adhc, "1 hopped selward", dc->getIdAndKind());
          ctalog.hop(gw, from, dc,  "lowest no theta buy");
        }

      // SECOND: try to back-track along the next unsold selgate, but capset en route

      else if ( gw = dc->lowestNotTransSel() )    // lowest transaction unsolved selgate
        {
          s_logger->repx(logga::adhc, "2 lowest not sold buy", gw->getIdAndKind());
          const std::string gate = gw->getIdAndKind();
          ctalog.event("select", "gateway " + gate, "lowest not transacted sel");

          if ( gw->getHash() == false )       // not fully capacitated
            {
              ctalog.capset(dc, gw);
              if ( dc->capset(step, gw, capacityMode) == false ) // set capacity bounds
                {
                  s_logger->repx(logga::adhc, "2 capset call failed", dc->getIdAndKind());

                  ctalog.alert("failure reason not available");
                  put << ctalog.output();    // see class 'EventRecorder'
                  s_logger->addSmartBlank(logga::dbug);
                  s_logger->putx(logga::dbug, put);
```

```
                return false;
              }
            s_logger->repx(logga::adhc, "2 capset call succeeded", "");
            put << ::formatLead("CAPSET", step, ++count)
                << " success : "
                << gw->getIdAndKind() << " capacity set by "
                << dc->getIdAndKind() << "\n";
            s_logger->putx(logga::adhc, put);

          }
        else
          {
            // no need to capacity set ('capset' would return
            // early in any case)
          }

        shared_ptr<DomainController> from = dc; // used in cta summary
        dc = gw->hop(dc);                       // buy-side hop
        s_logger->repx(logga::adhc, "2 hopped buyward", dc->getIdAndKind());
        ctalog.hop(gw, from, dc, "lowest no theta buy");

      }
    else
      {

        // THIRD: attempt to transolve

        ctalog.transolve(dc);
        if ( dc->transolve(step, capacityMode) == false )      // note side-effect
          {

#if   (XE_CTA == 1) // without relitigation

            s_logger->repx(logga::adhc, "3 transolve failed", dc->getIdAndKind());
            put << "  CTA (" << strStep << ")"
                << ": could not find a feasible solution, that is, unable"       << "\n"
                << "  to identify adequate or sufficiently flexible capacity"    << "\n"
                << "  domain : " << dc->getIdentifier()
                << "\n";
            s_logger->putx(logga::warn, put);

            ctalog.alert("could not find feasible solution");
            put << ctalog.output();        // see class 'EventRecorder'
            s_logger->addSmartBlank(logga::dbug);
            s_logger->putx(logga::dbug, put);

            return false;

#elif (XE_CTA == 2) // with hop-relitigation (compiles as of r4634 and r4773)

  // tofix: 16-Jul-2010: hop-relit: does 'looseflag' ever need to be reset
  // .. or made domain-specific? -- important only when this code is actioned

            // "looseness" test, used only for reporting
            if ( dc->lowestUnderCapBuy() &&     // lowest not-fully-capped buygate
                 dc->lowestSel() )              // lowest (meaning any) selgate
              {
                looseflag = true;               // potential problems ahead

                put << "  CTA (" << strStep << ")"
                    << ": identified a structure which cannot be totally\n"
                    << "  capacitiated and which may lead to a false infeasibility\n"
                    << "  domain : " << dc->getIdentifier()
                    << "\n";
                s_logger->putx(logga::warn, put);

                ctalog.alert("structure cannot be totally capacitated");
              }

            // hop-relitigate code
            if ( gw = dc->lowestUnderCapSel() ) // lowest not-fully-capped selgate
              {
                while ( shared_ptr<Gateway> temp = dc->lowestUnderCapSel() )
                  {
                    temp->unsetTransaction();   // set transaction status to not-sold
                    ctalog.unmark("sold", gw, dc, "me and all higher selgates");
                  }
                gw->unmarkStar();                   // set mark to false
                ctalog.unmark("star", gw, dc, "the me gateway referred to above");
                dc->capset(step, gw, capacityMode);  // set lower and upper capacities
                shared_ptr<DomainController> from = dc;   // used by 'ctalog'
```

```
                dc = gw->hop(dc);                    // hop right
                ctalog.hop(gw, from, dc, "hop right");
              }
            // no more hop-relitigate possibilities exist
            else
              {
                put << "  CTA (" << strStep << ")"
                    << ": could not find a feasible solution, that is, unable\n"
                    << "  to identify adequate or sufficiently flexible capacity\n"
                    << "  domain : " << dc->getIdentifier()
                    << "\n";
                s_logger->putx(logga::warn, put);

                if ( looseflag )
                  {
                    put << "  CTA (" << strStep << ")"
                        << ": capacity relitigation questionable"
                        << "\n";
                    s_logger->putx(logga::warn, put);
                  }
                else
                  {
                    put << "  CTA (" << strStep << ")"
                        << ": capacity relitigation robust"
                        << "\n";
                    s_logger->putx(logga::warn, put);
                  }

                ctalog.alert("unable to find feasible solution");
                put << ctalog.output();    // see class 'EventRecorder'
                s_logger->addSmartBlank(logga::dbug);
                s_logger->putx(logga::dbug, put);

                return false;
              }

#endif // macro 'XE_CTA'

          } // 'transolve' returned 'false'

        // FOURTH: 'transolve' returned 'true'

        else
          {
            s_logger->repx(logga::adhc, "4 transolve succeeded", dc->getIdAndKind());
            put << ::formatLead("TRANSOLVE", step, ++count)
                << " success : "
                << dc->getIdAndKind() << "\n";
            s_logger->putx(logga::adhc, put);

            // record transaction
            while ( gw = dc->lowestNotTransBuy() )
              {
                // next call naturally creates a 'foreach'
                // loop, moreover the gateway can recover its
                // own transaction solution from the solver
                shared_ptr<BuySide> temp = dynamic_pointer_cast<BuySide>(gw);
                const double duty = temp->recordTransaction();

                // reporting
                const std::string id = gw->getIdAndKind();
                s_logger->repx(logga::adhc, "4 transaction recorded", id);
                s_logger->repx(logga::adhc, "4 transaction", duty);
                ctalog.event("record", "transaction for " + gw->getIdAndKind(), duty);
              }

            // strictly optional block
            // YEEK 31 CODE (set by '--yeek')
            if ( xeona::yeek == 31 )
              {
                s_logger->repx(logga::dbug, "optional code begins, yeek", xeona::yeek);

                // actively undertake full capacitation --
                // strictly optional and computationally
                // expensive
                while ( gw = dc->lowestUnderCapSel() )
                  {
                    s_logger->repx(logga::adhc,
                                   "4 undercapacitated gate to capset",
                                   gw->getIdAndKind());
                    dc->capset(step, gw, capacityMode);
```

```
                    }

                  s_logger->repx(logga::dbug, "optional code ends, yeek", xeona::yeek);
                } // YEEK 31

              // move to next buygate domain
              if ( gw = dc->lowestNoStarBuy() )   // lowest not-star'ed buygate
                {
                  s_logger->repx(logga::adhc, "4 lowest no star buy", gw->getIdAndKind());
                  gw->markStar();                 // mark star
                  s_logger->repx(logga::adhc, "4 starred", gw->getStar());
                  shared_ptr<DomainController> from = dc;   // used in cta summary
                  dc = gw->hop(dc);               // sel-side hop
                  s_logger->repx(logga::adhc, "4 hopped selward", dc->getIdAndKind());
                  ctalog.hop(gw, from, dc, "lowest no star buy");
                }
              else if ( dc = origDomcons.pop() )  // simple next
                {
                  s_logger->repx(logga::adhc, "4 popped domcon", dc->getIdAndKind());
                  const std::string domcon = dc->getIdAndKind();
                  ctalog.event("pop", "pop next originating domcon " + domcon, "");
                  // CAUTION: do nothing is correct
                }
              else
                {
                  s_logger->repx(logga::adhc, "4 finishing", "");
                  put << "  CTA (" << strStep << ")"
                      << ": success: able to find a feasible solution set"
                      << "\n";
                  s_logger->putx(logga::warn, put);    // does not count as a "WARN"

                  ctalog.note("about to return success");
                  put << ctalog.output();              // see class 'EventRecorder'
                  s_logger->addSmartBlank(logga::dbug);
                  s_logger->putx(logga::dbug, put);

                  return true;
                }
            }

        }

    } // while true

} // function 'CtaSimple::captrans'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : gatesreg.cc
//  file-create-date : Sat 20-Feb-2010 19:48 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : find and register gateways / implementation
//  file-status      : work-in-progress
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/gatesreg.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "gatesreg.h"         // companion header for this file (place first)

#include "../b/teas.h"        // technical asset entity
#include "../b/node.h"        // LMP node entity
#include "../b/junc.h"        // demand split/join junction entity
#include "../b/gate.h"        // gateway entity
#include "../b/entity.h"      // entity base class plus lazy linking
#include "../b/domcon.h"      // domain controller entity
#include "../b/asop.h"        // asset operator entity

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <map>                // STL associative container
#include <string>             // C++ strings
#include <sstream>            // string-streams

//  CODE

class DomainController;

// ----------------------------------------------------------
//  FREE FUNCTION   : ::padInteger
// ----------------------------------------------------------
//  Description  : zero pad an integer
//  Role         : reporting calls
//  Techniques   : 'std::ostringstream' (rather than 'Boost::Format')
//  Status       : complete
// ----------------------------------------------------------

namespace
{
  std::string
  padInteger
  (const int integer,
   const int pad = 2)                        // note default
  {
    std::ostringstream oss;
    oss << std::setw(pad) << std::setfill('0') << integer;
```

```
    return oss.str();
  }
} // unnamed namespace

// ----------------------------------------------------------
//  FREE FUNCTION   : ::registerGateways
// ----------------------------------------------------------
//  Description  : record domain buy-gates / register both domains in gateways
//  Role         : called by 'xeona::registerGate'
//  Techniques   : traversal, 'dynamic_pointer_cast', 'std::map', 'std::map::find'
//  Status       : complete
//
//  Design notes
//
//      The following post-conditions should be fulfilled on exit:
//
//        - domain controllers have their 'd_randomBuygates' vectors filled
//        - gateways have their abutting domain controllers registered
//
//      Achieving these post-conditions requires a somewhat
//      involved matching process, based on the following:
//
//        - the "demander block" is in { gateway, demand
//          junction, lmp node, technical asset } (note that
//          gateways can connect to gateways to implement
//          arbitrage)
//
//        - knowledge of the demander block for each gateway
//          using domain-specific traversals of 'd_rankedSelgates'
//
//        - a constructed map of potential demander blocks and
//          their domain controllers
//
//      The matching process is straightforward, once the above
//      information has been obtained.
//
//      The code in this function is well documented and the
//      description just given should be self-evident.
//
//  Coding issues
//
//      [1] emptying an ostringsteam: 'str(const std::string s)'
//      will set the buffer contents to 's', see Lischner (2003
//      p653) and elsewhere
//
// ----------------------------------------------------------

namespace
{
  bool                                        // 'true' means runs to completion
  registerGateways
  (std::vector<shared_ptr<DomainController> > domains)
  {
    // initial reporting
    static logga::spLogger logger = logga::ptrLogStream();  // bind logger
    logger->repx(logga::adhc, "entering free function", "");

    // string stream for optional yeek report
    std::ostringstream put;

    // set return flag under the presumption of success
    bool returnFlag = true;

    // ---------------------------------
    //  1 - domain controllers
    // ---------------------------------

    // abandon if required
    if ( domains.empty() )
      {
        logger->repx(logga::dbug, "leaving free function early", "no domains");
        return returnFlag;
      }

    // ---------------------------------
    //  2 - prepare some 'std::map's
    // ---------------------------------

    // for convenience
    typedef std::map<shared_ptr<Gateway>, shared_ptr<DomainController> > mapGD_type;
    typedef std::map<shared_ptr<Gateway>, shared_ptr<Block>            > mapGB_type;
```

```
typedef std::map<shared_ptr<Block>  , shared_ptr<DomainController> > mapBD_type;

// declare some maps
mapGD_type mapGatesToDomains;            // entire system (hence both plurals)
mapGB_type mapGatesToBlocks;             // entire system (hence both plurals)
mapBD_type mapBlocksToDomains;           // entire system (hence both plurals)

// declare some iterators
mapGD_type::iterator pos1;
mapGB_type::iterator pos2;
mapBD_type::iterator pos3;

// ---------------------------------
//  3 - form the mappings
// ---------------------------------

// CAUTION: the 'xeona::mapCombine' calls need template
// instantiations in unit 'c/util1'

BOOST_FOREACH( shared_ptr<DomainController> domain, domains )
  {
    // capture the ranked selgate data
    mapGD_type gatesToDomain = domain->mapGatesToDomain();
    xeona::mapCombine(mapGatesToDomains, gatesToDomain);

    // drill upstream to demand block
    mapGB_type gatesToBlocks = domain->mapGatesToBlocks();
    xeona::mapCombine(mapGatesToBlocks, gatesToBlocks);

    // drill down to demand block
    mapBD_type blocksToDomain = domain->mapBlocksToDomain();
    xeona::mapCombine(mapBlocksToDomains, blocksToDomain);
  }

// ---------------------------------
//  4 - process the sel-side
// ---------------------------------

int lineCount1 = 0;                      // used for 'put' reporting
put << "  " << "gateway sel-side processing" << "\n";

for ( pos1  = mapGatesToDomains.begin();
      pos1 != mapGatesToDomains.end();
      ++pos1)
  {
    // obtain pointers
    shared_ptr<Gateway>          gate   = pos1->first;
    shared_ptr<DomainController> domain = pos1->second;

    // register domain on the sel-side
    shared_ptr<SelSide> sel = dynamic_pointer_cast<SelSide>(gate);
    sel->registerDomain(domain);

    put << "    "
        << "SEL-" << ::padInteger(++lineCount1) << "  "
        << std::setw(30+4+30) << std::left << gate->getIdAndKind() << " >  "
        << domain->getIdAndKind() << "\n";
  }

// ---------------------------------
//  5 - process the buy-side
// ---------------------------------

int lineCount2 = 0;                      // used for 'put' reporting
put << "\n"
    << "  " << "gateway buy-side processing" << "\n";

// loop gateways
for ( pos2  = mapGatesToBlocks.begin();
      pos2 != mapGatesToBlocks.end();
      ++pos2)
  {
    // obtain pointers
    shared_ptr<Gateway> gate  = pos2->first;
    shared_ptr<Block>   block = pos2->second;

    put << "    "
        << "BUY-" << ::padInteger(++lineCount2) << "  "
        << std::setw(30) << std::left <<  gate->getIdAndKind() << " >  "
        << std::setw(30) << std::left << block->getIdAndKind() << " >  ";
```

```
        // attempt to find block
        pos3 = mapBlocksToDomains.find(block);    // based on shared pointer comparison
        if ( pos3 == mapBlocksToDomains.end() )
          {
            logger->repx(logga::warn, "no domain match for block", block->getIdAndKind());
            returnFlag = false;

            put << "(no domain match)" << "\n";
          }
        else
          {
            // obtain pointer
            shared_ptr<DomainController> domain = pos3->second;

            // update 'd_randomBuygates'
            domain->addBuygate(gate);
            // register the domain on the buy-side
            shared_ptr<BuySide> buy = dynamic_pointer_cast<BuySide>(gate);
            buy->registerDomain(domain);

            put << domain->getIdAndKind() << "\n";
          }
      }

    // ---------------------------------
    //  6 - mop-up
    // ---------------------------------

    // additional reporting as appropriate
    // YEEK 20 CODE (set by '--yeek')
    if ( xeona::yeek == 20 || xeona::yeek == 1 )
      {
        logger->repx(logga::dbug, "additional reporting follows, yeek", xeona::yeek);
        logger->putx(logga::dbug, put);
      }

    // completion reporting
    logger->repx(logga::adhc, "leaving free function, return", returnFlag);

    // return
    return returnFlag;
  }

} // unnamed namespace

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::registerGates
// ------------------------------------------------------------
//  Description  : finalize reciprocal information for gateway and domain controllers
//  Role         : point of contact for registering gateways
//  Techniques   : 'Entity::getFullPopn', '::registerGateways', one pass protection
//  Status       : complete
//
//  Motivation
//
//      One might wonder about the complexity of the code in this
//      unit.  It would clearly be simpler to ask the modeler to
//      define 'd_randomBuygates'.  That however would add to
//      data redundancy and make 'xeona' models more brittle.
//      Hence the approach taken here.
//
//  Design notes
//
//      After confirming that there are full entities, this
//      function calls '::registerGateways.'  See the notes for
//      '::registerGateways' for details.
//
// ------------------------------------------------------------

namespace xeona
{
  bool                                      // 'true' means ran to completion
  registerGates()
  {
    // logger
    static logga::spLogger logger = logga::ptrLogStream();  // bind logger

    // one pass protection
    static unsigned passCount = 0;
    if ( ++passCount > 1 )
      {
```

```
        logger->repx(logga::warn, "repeat call, abandoning, pass count", passCount);
        return false;
      }
    else
      {
        logger->repx(logga::adhc, "entering free function, pass count", passCount);
      }

    // check for entities, otherwise return
    if ( Entity::getFullPopn() == 0 )
      {
        logger->repx(logga::info, "no full entities present", "abandoning search");
        return true;                        // considered a success
      }

    // active code
    std::vector<shared_ptr<DomainController> > domains = Entity::retDomains();
    const bool ret = ::registerGateways(domains);

    // return
    logger->repx(logga::dbug, "leaving free function, return", ret);
    return ret;
  }

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : ospinfo.cc
//  file-create-date : Mon 19-Oct-2009 11:04 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain mode interpretation / implementation
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/ospinfo.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "ospinfo.h"          // companion header for this file (place first)

#include "../c/util2.h"       // free functions which offer general utilities 2
#include "../c/util1.h"       // free functions which offer general utilities 1

#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro
#include <boost/format.hpp>   // printf style formatting

//  CODE

// ----------------------------------------------------------
//  CLASS           : DomainModeInfo
// ----------------------------------------------------------

DomainModeInfo::DomainModeInfo
(const xeona::DomainMode    mode,
 const std::string          shortMsg,
 const std::string          longMsg) :
  d_mode(mode),
  d_shortMsg(shortMsg),
  d_longMsg(longMsg),
  d_modeKind()
{
  // set mode kind string based on 'd_mode'
  if      ( d_mode  < xeona::e_modeNotSpecified ) d_modeKind = "out-of-range";
  else if ( d_mode == xeona::e_modeNotSpecified ) d_modeKind = "invalid";
  else if ( d_mode <= xeona::e_capacityModes    ) d_modeKind = "capacity";
  else if ( d_mode <= xeona::e_commitmentModes  ) d_modeKind = "commitment";
  else if ( d_mode <= xeona::e_maxAggregate     ) d_modeKind = "invalid";
  else                                            d_modeKind = "out-of-range";
}

std::string
```

```
DomainModeInfo::shortform() const
{
  std::ostringstream oss;
  oss << boost::format("%5d %s") % d_mode % d_shortMsg;
  return oss.str();
}

std::string
DomainModeInfo::longform() const
{
  std::ostringstream oss;
  oss << boost::format("%5d = %s : %s") % d_mode % d_modeKind % d_longMsg;
  return oss.str();
}

// ----------------------------------------------------------
//  CLASS           : DomainModeDatabase
// ----------------------------------------------------------

logga::spLogger DomainModeDatabase::s_logger = logga::ptrLogStream();

DomainModeDatabase::DomainModeDatabase() :
  d_infos()
{
  // initial reporting
  s_logger->repx(logga::xtra, "constructor call", "");

  // NOT-SET VALUE
  d_infos.push_back(DomainModeInfo(xeona::e_modeNotSpecified,    // 0
                                   "modeNotSpecified",
                                   "mode not specified"));
  // CONSTRAIN MODES
  d_infos.push_back(DomainModeInfo(xeona::e_usePresets,          // 1
                                   "usePresets",
                                   "use XEM data capacities, no run-time calculation"));
  d_infos.push_back(DomainModeInfo(xeona::e_withholdOkay,        // 2
                                   "withholdOkay",
                                   "withholding acceptable"));
  d_infos.push_back(DomainModeInfo(xeona::e_withholdBan,         // 4
                                   "withholdBan",
                                   "withholding banned, so normal technical capacity"));
  d_infos.push_back(DomainModeInfo(xeona::e_crisisOperation,     // 8
                                   "crisisOperation",
                                   "relax all further non-mandatory constraints"));
  // COMMITMENT MODES
  d_infos.push_back(DomainModeInfo(xeona::e_shortrunFin,         // 32
                                   "shortrunFin",
                                   "short-run financial cost minimization"));
  d_infos.push_back(DomainModeInfo(xeona::e_shortrunGhg,         // 64
                                   "shortrunGhg",
                                   "short-run GHG contribution minimization"));
  d_infos.push_back(DomainModeInfo(xeona::e_shortrunNox,         // 128
                                   "shortrunNox",
                                   "short-run NOx contribution minimization"));
  d_infos.push_back(DomainModeInfo(xeona::e_shortrunDep,         // 256
                                   "shortrunDep",
                                   "short-run depletable resource use minimization"));
  d_infos.push_back(DomainModeInfo(xeona::e_shortrunLuc,         // 512
                                   "shortrunLuc",
                                   "short-run land use minimization"));
  d_infos.push_back(DomainModeInfo(xeona::e_auctionLmp,          // 2048
                                   "auctionLmp",
                                   "locational marginal pricing auction"));
  d_infos.push_back(DomainModeInfo(xeona::e_adminMerit,          // 4096
                                   "adminMerit",
                                   "prescribed merit order"));
  d_infos.push_back(DomainModeInfo(xeona::e_adminFirst,          // 8192
                                   "adminFirst",
                                   "first feasible solution"));
}

std::string
DomainModeDatabase::getInfo
(const int domainMode) const
{
  BOOST_FOREACH( DomainModeInfo d, d_infos )
    {
      if ( d.d_mode == domainMode )
        {
          return d.longform();
        }
```

```
    }
  return "";
}

std::vector<std::string>
DomainModeDatabase::longform
(const int domainModes) const
{
  // initial reporting
  s_logger->repx(logga::adhc, "entering member function, modes", domainModes);

  // active code
  std::vector<std::string> buffer;
  std::vector<int> reds = xeona::geometricProgression(domainModes);
  BOOST_FOREACH( int i, reds )
    {
      if ( i == 0 ) continue;                // skip the zeros
      std::string buf = getInfo(i);
      if ( ! buf.empty() ) buffer.push_back(buf);
    }
  return buffer;
}

void
DomainModeDatabase::test
(std::ostringstream& oss) const
{
  BOOST_FOREACH( DomainModeInfo d, d_infos )
    {
      oss << "  " << d.longform() << "\n";
    }
}

// ------------------------------------------------------------
//  FREE FUNCTION   : xeona::infoDomainModeLong (twice)
// ------------------------------------------------------------

namespace xeona
{
  std::vector<std::string>
  infoDomainModeLong
  (const int domainModes)                    // can be non-pure
  {
    static DomainModeDatabase database;      // CAUTION: this variable is static
    return database.longform(domainModes);
  }

  std::string
  infoDomainModeLong
  (const int domainModes,                    // can be non-pure
   const int padding)
  {
    std::vector<std::string> interprets = infoDomainModeLong(domainModes);
    std::ostringstream oss;
    const std::string leftMargin(padding, ' ');
    BOOST_FOREACH( std::string s, interprets )
      {
        oss << leftMargin << s << "\n";
      }
    return oss.str();
  }

} // namespace 'xeona'

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : trav.cc
//  file-create-date : Wed 11-Feb-2009 13:42 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : domain graph traversal class / implementation
//  file-status      : complete
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/f/trav.cc $
//
//  GENERAL NOTES FOR THIS FILE

//  LOCAL AND SYSTEM INCLUDES

#include "trav.h"             // companion header for this file (place first)

#include "../b/domcon.h"      // domain controller entity
#include "../b/gate.h"        // gateway entity
#include "../c/util1.h"       // free functions which offer general utilities 1
#include "../c/util2.h"       // free functions which offer general utilities 2
#include "../a/logger.h"      // standard logging functionality (as required)
#include "../c/smart_ptr.h"   // toggle between Boost and TR1 smart pointers
#include ".././common.h"      // common definitions for project (place last)

#include <string>             // C++ strings
#include <sstream>            // string-streams

#include <boost/foreach.hpp>  // BOOST_FOREACH iteration macro

//  CODE

// STATIC DEFINITIONS - with and without explicit initialization

logga::spLogger TravDepthFirst::s_logger = logga::ptrLogStream();

// CREATORS

// -----------------------------------------------------------
//  MEMBER FUNCTION : TravDepthFirst
// -----------------------------------------------------------
//  Description  : constructor
//  Role         : services the 'Overseer' singleton
//  Techniques   : constructor contains internal call to DFS algorithm
//  Status       : complete
//
//  Design notes
//
//      Construction automatically invokes the DFS algorithm, due
//      to the call to private member function 'depthFirstTrav'.
//
//      The subsequent data recovery calls can simply return
//      the information recovered earlier.
//
// -----------------------------------------------------------
```

```
TravDepthFirst::TravDepthFirst
(const std::vector<shared_ptr<DomainController> >& domcons) :  // order significant
   d_originatingDomcons(domcons),
   d_allDomcons(),                          // empty vector
   d_componentCount(0)                      // initialize to zero
{
  s_logger->repx(logga::xtra,
                 "constructor call, orig domcon count",
                 d_originatingDomcons.size());

  depthFirstTrav();                         // sole call to this utility function
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : ~TravDepthFirst
// ----------------------------------------------------------

TravDepthFirst::~TravDepthFirst()
{
  s_logger->repx(logga::adhc, "destructor call", "");
}

// ACCESSORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : getOrigDomconsCount
// ----------------------------------------------------------

int                                         // number of originating domains
TravDepthFirst::getOrigDomconsCount() const
{
  return d_originatingDomcons.size();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getAllDomconsCount
// ----------------------------------------------------------

int                                         // total number of domains
TravDepthFirst::getAllDomconsCount() const
{
  return d_allDomcons.size();
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getComponentsCount
// ----------------------------------------------------------

int                                         // number of maximally connected sub-graphs
TravDepthFirst::getComponentsCount() const
{
  return d_componentCount;
}

// ----------------------------------------------------------
//  MEMBER FUNCTION : getAllDomcons
// ----------------------------------------------------------

bool
TravDepthFirst::getAllDomcons
(std::vector<shared_ptr<DomainController> >& allDomcons) const // note non-const reference
{
  if ( ! allDomcons.empty() )
    {
      s_logger->repx(logga::warn, "load up container not empty", d_allDomcons.size());
      s_logger->repx(logga::xtra, "abandoning function without action", "");
      return false;
    }
  allDomcons = d_allDomcons;                // copy assignment to non-const reference
  return true;
}

// MANIPULATORS

// ----------------------------------------------------------
//  MEMBER FUNCTION : callDomains (with 'step' pass thru)
// ----------------------------------------------------------
//  Description  : traverse all domains and call 'func' on each domain
//  Role         : used by 'Overseer' to launch 'constrain' calls and similar
//  Techniques   : 'func' pointer-to-member function argument
```

```
//  Status       : complete
//
//  Design notes (which also mostly apply to the other overloads)
//
//      Typical applications
//
//          This is a matter for client code but some typical
//          applications are listed here nonetheless:
//
//          - horizon initialization traversal
//          - interval initialization traversal
//          - interval finalization traversal
//          - horizon finalization traversal
//
//      Depth first search on construction
//
//          The underpinning depth first search (DFS) is
//          undertaken on construction.  An update can be achieve
//          by invoking a shared pointer delete/new cycle,
//          namely:
//
//              dfs.reset(new TravDepthFirst(rankedOrigDomains))
//
//      Usage
//
//          The pointer-to-member function argument is explained
//          in the overarching documentation for class
//          'TravDepthFirst' in the header.
//
//      Pointer function call syntax
//
//          The following syntax should work for raw pointers,
//          but would NOT with Boost.Smart_ptr smart pointers:
//
//              (dc->*func)(step)
//
//          The g++ 4.1.2 compiler chokes thus -- see the code
//          proper for a simple work around:
//
//              error: no match for 'operator->*' in 'dc ->* func'
//
// -------------------------------------------------------

const int                                   // number of visited domains
TravDepthFirst::callDomains
(funcPtrStep func,                          // see typedef in header
 const int   step)                          // duly passed thru to the 'func' call
{
  int count = 0;
  BOOST_FOREACH( shared_ptr<DomainController> dc, d_allDomcons )
    {
      ++count;
      ((*dc).*func)(step);                  // CAUTION: syntax is correct, see above
    }
  return count;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : callDomains (without 'step' pass thru)
// -------------------------------------------------------

const int                                   // number of visited domains
TravDepthFirst::callDomains
(funcPtrVoid func)                          // see typedef in header
{
  int count = 0;
  BOOST_FOREACH( shared_ptr<DomainController> dc, d_allDomcons )
    {
      ++count;
      ((*dc).*func)();                      // CAUTION: syntax is correct, see above
    }
  return count;
}

// -------------------------------------------------------
//  MEMBER FUNCTION : callDomains (without 'step' pass thru)
// -------------------------------------------------------

const int                                   // number of visited domains
TravDepthFirst::callDomains
(funcPtrStepCs3 func,                       // see typedef in header
```

```
 const int      step,
 CostSet&       var,
 CostSet&       fix,
 CostSet&       emb)
{
  int count = 0;
  BOOST_FOREACH( shared_ptr<DomainController> dc, d_allDomcons )
    {
      ++count;
      ((*dc).*func)(step, var, fix, emb);    // CAUTION: syntax is correct, see above
    }
  return count;
}

// UTILITY FUNCTIONS

// -----------------------------------------------------------
//  MEMBER FUNCTION : depthFirstTrav
// -----------------------------------------------------------
//  Description  : traverse entire domain graph given only the originating domains
//  Role         : support for custom domain graph traversals
//  Techniques   : depth first search (DFS) using a non-recursive labeling scheme
//  Status       : complete but not tested
//
//  Design notes
//
//      Originating domains
//
//          Note that the 'd_originatingDomcons' originating
//          domains list can be interpreted as representing the
//          prioritized selgates from a single hypothetical
//          originating domain.  Indeed some DFS methods begin by
//          creating such a "genesis" node.
//
//      Push/pop buygate calls
//
//          The buygate ordering is arbitrary and -- unlike
//          selgates -- no priority ranking is required.  In this
//          case, the ordering is determined by the visitation
//          process.
//
//      Tilde and hyphen flags
//
//          The 'tilde' gateway and 'hyphen' domain flags are
//          reserved for this procedure.  For completeness, the
//          other gateway flags comprise 'theta' 'star' and
//          'hash' and are for use by the captrans algorithm
//          (CTA).
//
//      Topological sort order
//
//          One side effect of a DFS search is that the resulting
//          visitation order is topologically sorted.
//
//      Graph components (unconnected sub-graphs) count
//
//          The DFS method can also identify and count the number
//          of individual unconnected sub-graphs.  The result is,
//          at the time of writing, logged at the level of
//          'logga::info', but is not written to the XEM file as
//          output data (that however is a matter for the client
//          code and may change).
//
// -----------------------------------------------------------

bool
TravDepthFirst::depthFirstTrav()
{
  s_logger->repx(logga::dbug, "entering member function", "");

  // ---------------------------------
  //  input data integrity check
  // ---------------------------------

  if ( d_originatingDomcons.empty() )        // client supplied data, so may be empty
    {
      if ( xeona::releaseStatus == true )    // complain more
        {
          s_logger->repx(logga::warn, "no originating domain controllers", "");
        }
      else
```

```
      {
        s_logger->repx(logga::rankNoData, "no originating domain controllers", "");
      }
    return true;                            // not considered an error
  }

// ---------------------------------
//  multiple invocation protection
// ---------------------------------

// no need to repeat the traversal once done

static bool firstInvocation = true;         // static member
if ( firstInvocation == false )             // subsequent calls abandoned here
  {
    s_logger->repx(logga::dbug, "repeat invokation", "");
    s_logger->repx(logga::xtra, "abandoning function without action", "");
    return true;
  }
firstInvocation = false;                    // update status

// ---------------------------------
//  member data integrity check
// ---------------------------------

if ( ! d_allDomcons.empty() )               // should never get here
  {
    const int nonzero = d_allDomcons.size();
    s_logger->repx(logga::warn, "domain controllers vector not empty", nonzero);
    s_logger->repx(logga::xtra, "abandoning traversal without action", "");
    return false;
  }

// ---------------------------------
//  preamble
// ---------------------------------

// for logging
std::ostringstream put;

// load the domain controllers vector into a more convenient popable container
SmartPtrPopper<DomainController> origDomcons;   // simple popable container class
origDomcons.load(d_originatingDomcons);         // shared pointers elements copied over

// declare some entity variables -- the shared pointers are naturally empty
shared_ptr<DomainController> dc;
shared_ptr<Gateway>          gw;

// declare some administrative variables
bool crossEdgeFlag;                         // 'true' if a cross edge was found

// ---------------------------------
//  starting position
// ---------------------------------

dc               = origDomcons.pop();       // grab first originating domain, must exist
d_componentCount = 1;                       // now unity, at least one vertex present [1]
crossEdgeFlag    = false;                   // edge which connects two spanning trees [2]

// [1] a component of a graph is a maximally connected
// sub-graph -- in lay terms, this is the largest possible
// navigable subset of the original graph
//
// [2] that is, an edge which connects the spanning trees which
// derive from any two (necessarily distinct) originating
// domains

// ---------------------------------
//  main loop
// ---------------------------------

// CAUTION: the single "=" (copy assignment operator) in the
// various selection statements ('if' and perhaps 'switch' ) is
// normally correct

// [3] (see below) a hyphen represents a domain visited flag
// [4] (see below) a tilde represents a selgate visited flag

s_logger->repx(logga::dbug, "* DFS loop starts", "");

while ( true )                              // a clear idiom but not the most compact
```

```
    {
      s_logger->repx(logga::adhc, "- loop start with domcon", dc->getIdAndKind());

      // ZERO: add the current domain if not marked, otherwise
      // note the presence of a cross edge

      if ( dc->getHyphen() )                // [3]
        {
          s_logger->repx(logga::adhc, "0 domcon with hyphen", dc->getIdAndKind());
          crossEdgeFlag = true;
          s_logger->repx(logga::adhc, "0 cross-edge set", crossEdgeFlag);
        }
      else
        {
          s_logger->repx(logga::adhc, "0 domcon without hyphen", dc->getIdAndKind());
          dc->markHyphen();                 // duly mark
          s_logger->repx(logga::adhc, "0 domcon now hyphened", dc->getHyphen());
          d_allDomcons.push_back(dc);       // capture by copying in smart pointer
          s_logger->repx(logga::adhc, "0 all domcon added, size", d_allDomcons.size());
        }

      // FIRST: attempt to head deep

      if ( gw = dc->lowestNoTildeSel() )    // lowest not-tilde'd selgate [4]
        {
          s_logger->repx(logga::adhc, "1 lowest no tilde sel", gw->getIdAndKind());
          gw->markTilde();                  // mark tilde
          s_logger->repx(logga::adhc, "1 gateway tilded", gw->getIdAndKind());
          dc = gw->hop(dc);                 // buy-side hop
          s_logger->repx(logga::adhc, "1 hopped buyward", dc->getIdAndKind());
          dc->pushTilde(gw);                // push the gateway, also collect the buygate
          s_logger->repx(logga::adhc, "1 domain push tilde", gw->getIdAndKind());
        }

      // SECOND: attempt to back-track

      else if ( gw = dc->popTilde() )       // pop earlier push
        {
          s_logger->repx(logga::adhc, "2 popped tilde", gw->getIdAndKind());
          dc = gw->hop(dc);                 // sel-side hop
          s_logger->repx(logga::adhc, "2 hopped selward", dc->getIdAndKind());
        }

      // THIRD: move to next buygate domain

      else if ( dc = origDomcons.pop() )    // simple next
        {
          // CAUTION: do nothing active is correct
          if ( crossEdgeFlag == false ) ++d_componentCount;
          crossEdgeFlag = false;            // reset flag
          s_logger->repx(logga::adhc, "3 next domcon", dc->getIdAndKind());
        }

      // FOURTH: termination

      else
        {
          s_logger->repx(logga::adhc, "4 termination", "");
          break;                            // from the 'while' block
        }
    } // while true

  s_logger->repx(logga::adhc, "* DFS loop finished", "");

  // ---------------------------------
  //  traversal reporting
  // ---------------------------------

  const std::string notes =
    "    notes: the originating domains are listed by decreasing importance\n"
    "           a component is an energy island, more formally, each largest"
    " possible navigable subset of the original domain graph";
  put << "  DFS (depth-first search): success: traversal complete"       << "\n"
      << "    originating domains count     : " << getOrigDomconsCount() << "\n"
      << "    all domains count             : " << getAllDomconsCount()  << "\n"
      << "    domain graph components count : " << getComponentsCount()  << "\n"
      << "\n"
      << notes                                                           << "\n";
  s_logger->repx(logga::dbug, "in-depth reporting follows", "");
  s_logger->putx(logga::dbug, put);
```

```
  // ---------------------------------
  //    flag resetting
  // ---------------------------------

  // reset the tilde and hyphen flags as a precaution

  BOOST_FOREACH( shared_ptr<DomainController> dc, d_allDomcons )
    {
      dc->resetTildeSelgates();              // 'DomainController's continue the resetting
      dc->resetHyphen();
    }

  // ---------------------------------
  //  abandon if yeek 23
  // ---------------------------------

  // additional action as appropriate
  // YEEK 23 CODE (set by '--yeek')
  if ( xeona::yeek == 23 )
    {
      std::string msg;
      msg += "at the end of function '";
      msg += __func__;
      msg += "'";
      s_logger->repx(logga::xtra, "will throw xeona::yeek_abandon", xeona::yeek);
      throw xeona::yeek_abandon(msg);
    }

  // ---------------------------------
  //    return success
  // ---------------------------------

  s_logger->repx(logga::dbug, "normal exiting of member function", "");
  return true;                               // should always get here (yeek 23 aside)
}

//  end of file
```

```
//  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT
//
//  file-create-name : main.cc
//  file-create-date : Mon 16-Apr-2007 11:45 UTC
//  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
//
//  file-role        : contains 'main' function, includes command-line parsing / main
//  file-status      : working
//  file-release-tag :
//
//  LEGAL NOTICE
//
//  Software  : This file is part of the source code for the xeona energy
//              systems modeling environment.
//  License   : This software is distributed under the GNU General Public
//              License version 3, a copy of which is provided in the text
//              file LICENSE_GPLv3.
//  Warranty  : There is no warranty for this software, to the extent permitted
//              by applicable law.  Refer to the license for further details.
//  Copyright : This software is copyright (c) 2007 - 2010 Robbie Morrison.
//  Request   : The software is distributed with the request that you forward
//              any modifications you make to the xeona project for possible
//              inclusion in the main codebase.
//
//  PROJECT CONTACT
//
//  Robbie Morrison
//  Institute for Energy Engineering
//  Technical University of Berlin
//  Marchstrasse 18, D-10587 Berlin, Germany
//  Email: robbie@actrix.co.nz
//
//  SVN VERSION CONTROL
//
//  $Author: robbie $
//  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/main.cc $
//
//  GENERAL NOTES FOR THIS FILE
//
//  This is the main file for the xeona application.

//  LOCAL AND SYSTEM INCLUDES

#include "a/logger.h"         // standard logging functionality (as required) [1]

// [1] CAUTION: "a/logger.h" should be first -- there were early
// problems with the logger destructor '~Logger' but that could
// well have been due to the used of 'exit()' calls -- which no
// longer applies

#include "c/xeona_ptr.h"      // remappable counted pointer which mimics shared_ptr
#include "c/simcall.h"        // main simulation call
#include "c/inbuilt.h"        // compiled-in test file generation
#include "a/yeek.h"           // yeek (for running extra code) value interpretation
#include "a/xemopt.h"         // skeleton xem model generator
#include "a/xedocs.h"         // processing of 'xedoc' entity documentation
#include "a/exitstat.h"       // exit status database
#include "a/exapp.h"          // application exception classes
#include "a/appinfo.h"        // generates version info, splash screens, and so forth

#include "./common.h"         // common definitions for project (place last)

#include <iostream>           // standard io
#include <sstream>            // string-streams
#include <stdexcept>          // standard exception classes, runtime_error()
#include <string>             // C++ strings

#include <cstdio>             // C-style io, remove(), EOF, perror()

#include <unistd.h>           // POSIX sleep(), usleep(), access(), chown()

#include <boost/algorithm/string.hpp>   // string recasing, trimming, splitting
#include <boost/filesystem.hpp>         // path objects, iterators, useful operations
#include <boost/format.hpp>             // printf style formatting
#include <boost/program_options.hpp>    // command-line parser and more
#include <boost/weak_ptr.hpp>           // Boost weak pointer

// CAUTION: Boost.Program_options needs a -lboost_program_options linking directive

//  CODE
```

```
// ------------------------------------------------------------
//  MAIN FUNCTION   : main
// ------------------------------------------------------------

int
main
(int   argc,                                  // CAUTION: do not use 'const' here
 char* argv[])                                // CAUTION: do not use 'const' here
{
  // ---------------------------------
  //  preamble
  // ---------------------------------

  // declare some returns, these values should ALWAYS be overwritten
  xeona::SimRet sret = xeona::e_statusNotKnown;        // return enum from 'simulate'
  int returnStatus   = xeona::ret_not_overwritten;     // application return status

  // main function logger
  logga::spLogger logger = logga::ptrLogStream();      // bind logger
  logger->repx(logga::dbug, "start of main", "~~~~~~~~~~");

  // grab initial current directory
  const boost::filesystem::path initial = boost::filesystem::current_path();  // [1]

  // [1] this functionality used to be provided by
  // 'boost::filesystem::initial_path' but that function was
  // withdrawn with Boost.Filesystem version 3 under 1.44.0

  // create application information object
  AppInfo appInfo(argc, argv, initial);      // pass thru command-line information

  // declare variables which may have been set on the command-line
  unsigned beepMode       = 0;               // changeable via --beep
  unsigned reportLevel    = 0;               // changeable via --report
  unsigned simulationMode = 0;               // changeable via --mode
  unsigned inbuiltSteps   = 0;               // changeable via --inbuilt
  unsigned exitTripLevel  = 0;               // changeable via --exittrip
  unsigned outputQuery    = 1000;            // set via --output, value is outside [0,255]
  unsigned yeekValue      = 0;               // see via --yeek
  std::string cmdLineModelName;              // single word or several if in quotes
  std::string logTrigger;                    // single word without whitespace for --watch
  std::string xedocClass;                    // single word without whitespace for --class
  std::string xemPart = "invalid";           // one of three supported terms for --xem

  // declare required default values (others are also set in "common.h")
  xeona::SimKind simKind = xeona::e_fullRun; // simulation mode (as enum)

  // declare some flags
  bool flag_generateInbuiltFile    = false;  // reset by --inbuilt
  bool flag_refreshModelIfPossible = false;  // reset by --guard

  // =================================
  //  parse command-line using Boost
  // =================================

  //  Boost.Program_options parses as follows:
  //
  //   * options are displayed and processed in the order given
  //     under 'add_options' function
  //
  //   * the first error encountered prevents further parsing

  namespace po = boost::program_options;     // for convenience

  logger->repx(logga::dbug, "command-line parsing, argc", argc);

  std::ostringstream ossBeepMsg;             // for use in --help message
  ossBeepMsg
    << "set beep behavior "
    << "0-2"
    << " (silent to noisy) (default "
    << boost::format("%d") % xeona::beepModeDefault         // set in 'common.cc'
    << ")";
  std::string sBeepMsg = ossBeepMsg.str();
  const char* cbeepMsg = sBeepMsg.c_str();

  std::ostringstream ossReportMsg;           // for use in --help message
  ossReportMsg
    << "set report level "
    << boost::format("%d-%d") % logga::yeek % (logga::adhc + 1)
    << " (nil to verbose) (default "
```

```
    << boost::format("%d") % xeona::reportLevelDefault      // set in 'common.cc'
    << ")";
  std::string sReportMsg = ossReportMsg.str();
  const char* creportMsg = sReportMsg.c_str();

  std::ostringstream ossModeMsg;             // for use in --help message
  ossModeMsg
    << "set run mode "
    << boost::format("%d-%d") % xeona::e_hollowCall % xeona::e_fullRun
    << " (reduced to normal) (default "
    << boost::format("%d") % xeona::e_fullRun
    << ")";
  std::string sModeMsg = ossModeMsg.str();
  const char* cmodeMsg = sModeMsg.c_str();

  std::ostringstream ossExitTripMsg;          // for use in --help message
  ossExitTripMsg
    << "set nonzero exit trip to ARG level logs (default "
    << boost::format("%d") % xeona::exitTripLevelDefault    // set in 'common.cc'
    << ")"
    << "\n";
  std::string sExitTripMsg = ossExitTripMsg.str();
  const char* cexitTripMsg = sExitTripMsg.c_str();

  // ---------------------------------
  //  command-line 'try' block
  // ---------------------------------

  try
    {
      // specify the options displayed under --help, note use of newlines
      const unsigned optionsWidthDefault = po::options_description::m_default_line_length;
      const unsigned optionsWidth        = 100;   // effective iff greater than default
      logger->repx(logga::dbug, "program_options default width", optionsWidthDefault);
      logger->repx(logga::dbug, "program_options given width"  , optionsWidth);
      const char* cOptionsMsg =
        "options (processed in order shown, can be abbreviated)";
      po::options_description desc(cOptionsMsg, optionsWidth);
      desc.add_options()
        ("help",     "display command-line options and quit")
        ("usage",    "display longer usage message and quit")
        ("output",
         po::value<unsigned>(&outputQuery), // default value not provided
         "describe application return code ARG and quit\n")

        ("legal",    "display copyright notice and license and quit")
        ("svn",      "display embedded source code revision number and quit")
        ("version",  "display version details in structured form and quit\n")

        ("data",  "display xem file data rules and quit")
        ("class",
         po::value<std::string>(&xedocClass),
         "display fields for entity class regex ARG and quit")
        ("xem",
         po::value<std::string>(&xemPart),
         "display ARG head|mand|tail|comb xem file parts and quit\n")

        ("quiet",    "silence all output except the above (release builds)\n")

        ("beep",
         po::value<unsigned>(&beepMode),
         cbeepMsg)                           // CAUTION: must be a const C-string
        ("report",
         po::value<unsigned>(&reportLevel),
         creportMsg)
        ("nodata",   "empty data fields generate warnings not debugs")
        ("jumpy",    "range failures and such generate warnings not debugs")
        ("watch",
         po::value<std::string>(&logTrigger),
         "report fully from sources matching ARG regex")
        ("exittrip",
         po::value<unsigned>(&exitTripLevel),
         cexitTripMsg)

        ("mode",
         po::value<unsigned>(&simulationMode),
         cmodeMsg)
        ("pepper",   "reseed random number generators using time() return")
        ("again",    "continue when choked or hard problems encountered")
        ("krazy",    "omit defensive coding and try to run to failure")
        ("yeek",
```

```
     po::value<unsigned>(&yeekValue),
     "run non-standard code based on ARG (changeable)")
    ("zero",     "disable close-to-zero solver output rounding\n")

    ("inbuilt",
     po::value<unsigned>(),
     "generate and run inbuilt test model using ARG steps")
    ("guard",    "regenerate xem file from guard file if present")
    ("tidy",     "shallow reformat xem file (alias --report 3 --mode 3)")
    ; // final semicolon

  // specify options not displayed under --help
  po::options_description hidden("hidden options");
  hidden.add_options()
    ("filename",                        // CAUTION: must be processed after "inbuilt"
     po::value<std::string>(&cmdLineModelName),
     "model name");

  // combine the above
  po::options_description cmdline_options;
  cmdline_options.add(desc).add(hidden);

  // specify treatment of stand-alone argument
  po::positional_options_description pod;
  pod.add("filename", -1);

  // final admin
  po::variables_map vm;
  po::store
    (po::command_line_parser(argc, argv).     // CAUTION: not 'parse_command_line'
     options(cmdline_options).positional(pod).run(), vm);
  po::notify(vm);                             // CAUTION: necessary call

  // ---------------------------------
  //  process command-line options
  // ---------------------------------

  if ( vm.count("help") )
    {
      logger->setReportLevel(0);         // reduce further logging to a minimum
      std::clog << std::flush;
      std::cout << desc;                 // Boost.Program_options
      return xeona::ret_message;
    }

  if ( vm.count("usage") )
    {
      logger->setReportLevel(0);         // reduce further logging to a minimum
      std::clog << std::flush;
      std::ostringstream ssDesc;
      ssDesc << desc;                    // to make 'desc' available to AppInfo
      std::cout << appInfo.showHelp(ssDesc.str());
      return xeona::ret_message;
    }

  if ( vm.count("output") )
    {
      logger->setReportLevel(0);         // reduce further logging to a minimum
      std::clog << std::flush;
      ExitStatus exitstatus;             // accept default not-present message
      std::string interpretation;        // fill by reference
      if ( exitstatus(outputQuery, interpretation) )   // 'true' if present
        {
          std::cout << interpretation << "\n";
          std::cout << std::flush;
        }
      return xeona::ret_message;
    }

  if ( vm.count("legal") )
    {
      std::clog << std::flush;
      std::cout << appInfo.showLegalMessage();
      return xeona::ret_message;
    }

  if ( vm.count("svn") )
    {
      std::clog << std::flush;
      std::cout << appInfo.getSvnRevision() << std::endl;
      return xeona::ret_message;
```

```
    }

  if ( vm.count("version") )
    {
      std::clog << std::flush;
      std::cout << appInfo.showParse();
      return xeona::ret_message;
    }

  if ( vm.count("class") && ! xedocClass.empty() )
    {
      if ( boost::find_token(xedocClass, boost::is_space()) )
        {
          std::ostringstream oss;
          oss << "'class' argument cannot contain whitespace: "
              << xedocClass;
          throw std::domain_error(oss.str()); // see Lischner (2003 p658)
        }
      else if ( xedocClass == "+" )           // not expanded by the shell
        {
          std::string plusStr = "\"" + xedocClass + "\"";
          logger->repx(logga::info, "class list requested, argument", plusStr);
          std::string classes = "";
          Xedocs xedocs;
          const int numberClasses = xedocs.dumpClassNames(classes);
          std::clog << std::flush;
          if ( numberClasses == 0 )
            {
              std::cout << appInfo.getProgramName()
                        << ": entity class database not refreshed at compile-time"
                        << " (probably built with 'make' call and not 'mach' script)"
                        << std::endl;
            }
          else
            {
              std::cout << "\n"
                        << classes
                        << std::endl;
            }
          return xeona::ret_message;
        }
      else
        {
          std::string regexStr = "\"" + xedocClass + "\"";
          logger->repx(logga::info, "xedocs requested for regex", regexStr);
          std::string xedoc = "";
          Xedocs xedocs;
          Xedocs::FindStatus findStatus = xedocs.findXedocForRegex(xedocClass, xedoc);
          std::clog << std::flush;

          switch ( findStatus )
            {
            case Xedocs::e_classFound:
              {
                std::cout << "\n" << xedoc << "\n" << std::endl;
                return xeona::ret_success;
              }
            case Xedocs::e_classNotFound:
              {
                std::cout << "case-sensitive regex search for '"
                          << xedocClass
                          << "' failed (try --class + for full listing)"
                          << std::endl;
                return xeona::ret_noclass;
              }
            case Xedocs::e_emptyDatabase:
              {                           // CAUTION: local block necessary
                std::cout << appInfo.getProgramName()
                          << ": entity class database not refreshed at compile-time"
                          << " (probably built with 'makefile' make call"
                          << " and not 'mach' build script)"
                          << std::endl;
                return xeona::ret_noclass;
              }
            default:
              {
                std::clog << "** coding error 01 in source file " << __FILE__
                          << std::endl;
                return xeona::ret_fail;
              }
            } // switch
```

```
            }
        }

      if ( vm.count("xem") )
        {
          logger->flush();
          Xem xem(std::cout,                // print to stream
                  xeona::svnRev,            // svn revision
                  45);                      // alignment tab (typically 45 or 50)
          if      ( xemPart == "head" )
            {
              xem.head();
              xem.blank();
            }
          else if ( xemPart == "mand" )
            {
              xem.mand(6);                  // number of steps
              xem.blank();
            }
          else if ( xemPart == "tail" )
            {
              xem.tail();
              xem.blank();
            }
          else if ( xemPart == "comb" )
            {
              xem.head();
              xem.mand(6, "(model goes here)");
              xem.tail();
            }
          else if ( xemPart == "string-delim" )
            {
              std::cout << xeona::modelStringDelim << std::endl;
            }
          else if ( xemPart == "disable-char" )
            {
              std::cout << xeona::modelDisableChar << std::endl;
            }
          else
            {
              std::ostringstream oss;
              oss << "'xem' argument not supported: " << xemPart;
              throw std::domain_error(oss.str());      // see Lischner (2003 p658)
            }
          xem.flush();
          logger->repx(logga::dbug, "--xem option code complete", xemPart);
          return xeona::ret_message;
        }

      if ( vm.count("quiet") )
        {
#if 1
// #if defined(__linux__) || defined(__unix__)

          // simply close 'stdout' and 'stderr'

          // CAUTION: the following might not cross-compile to
          // Windows -- in which case place under conditional
          // compilation

          // NOTE: there will be some leakage (around 16 console
          // lines) for non-release builds because of logging
          // output before this code is reached.

          std::cout << std::flush;
          std::clog << std::flush;

          if ( close(STDOUT_FILENO) == -1 )    // macro defined in <unistd.h>
            {
              std::clog << "** quiet code close stdout failed" << std::endl;
              perror("perror: failed to close stdout / system message");
            }
          if ( close(STDERR_FILENO) == -1 )    // macro defined in <unistd.h>
            {
              std::clog << "** quiet code close stderr failed" << std::endl;
              perror("perror: failed to close stderr / system message");
            }

          std::cout << std::flush;
          std::clog << std::flush;
#else
```

```
          logger->repx(logga::info, "option --quiet not yet implemented", "");
#endif // 0
        }

      if ( vm.count("data") )
        {
          std::clog << std::flush;
          std::cout << appInfo.showXemRules();
          return xeona::ret_message;
        }

      // CAUTION: the options above return, the options below do not

      if ( vm.count("again") )
        {
          const bool was = xeona::again;               // set 'false' in 'common.cc'
          if ( xeona::nopro ) xeona::again = false;    // toggle under '--again'
          else                xeona::again = true;
          std::ostringstream oss;
          oss << std::boolalpha;                       // use labels
          oss << was << " > " << xeona::again;
          logger->repx(logga::dbug, "xeona::again reset from > to", oss.str());
        }

      if ( vm.count("beep") )
        {
          switch ( beepMode )
            {
            case 2:
              logger->setBeepOnOrAbove(logga::warn);   // logga::yeek excluded
            case 1:
              logger->setBeepOnCompletion();
              logger->enableBeeping();
            case 0:                                    // do nothing
              break;
            default:
              {
                logger->repx(logga::dbug, "beep mode not supported", beepMode);
                std::ostringstream oss;
                oss << "'beep' " << beepMode << " is not supported";
                throw std::domain_error(oss.str());    // see Lischner (2003 p658)
              }
              break;
            }
        }

      if ( vm.count("exittrip") )
        {
          switch ( exitTripLevel )
            {
            case logga::yeek:
            case logga::kill:
            case logga::warn:
            case logga::info:
            case logga::dbug:
            case logga::xtra:
            case logga::adhc:
              {                                // CAUTION: keep any variables local
                logger->repx(logga::dbug,
                             "exit trip on log level set or reset",
                             exitTripLevel);
              }
              break;
            default:
              {
                logger->repx(logga::dbug,
                             "exit trip on log level unsupported",
                             exitTripLevel);
                std::ostringstream oss;
                oss << "exit trip on log level " << exitTripLevel << " is not supported";
                throw std::domain_error(oss.str());    // see Lischner (2003 p658)
              }
              break;
            }
        }

      // CAUTION: the following would not compile:
      // << vm["filename"].as<std::vector<std::string> >()
      // although the string version would

      if ( vm.count("filename") )            // quite possibly an empty string
```

```
    {
      appInfo.setCommandLineModelName(cmdLineModelName);     // just for reporting
    }

  if ( vm.count("guard") )
    {
      if ( flag_generateInbuiltFile )    // --inbuilt already processed
        {
          std::string msg = "cannot use --inbuilt and --guard together";
          throw std::invalid_argument(msg); // see Lischner (2003 p658)
        }
      flag_refreshModelIfPossible = true;
    }

  if ( vm.count("inbuilt") )
    {
      if ( cmdLineModelName.empty() )
        {
          cmdLineModelName = xeona::modelInbuiltDefault;
        }
      inbuiltSteps = vm["inbuilt"].as<unsigned>();

      if ( inbuiltSteps < 2 )
        {
          logger->repx(logga::dbug, "steps below 2 not supported", inbuiltSteps);
          std::ostringstream oss;
          oss << "'inbuilt' " << inbuiltSteps
              << " is not legal (must be 2 steps or more)";
          throw std::domain_error(oss.str());      // see Lischner (2003 p658)
        }
      flag_generateInbuiltFile = true;   // flag used to delay code execution
    }

  if ( vm.count("krazy") )
    {
      const bool was = xeona::nopro;               // set 'false' in 'common.cc'
      if ( xeona::nopro ) xeona::nopro = false;    // toggle under '--krazy'
      else                xeona::nopro = true;
      std::ostringstream oss;
      oss << std::boolalpha;                       // use labels
      oss << was << " > " << xeona::nopro;
      logger->repx(logga::dbug, "xeona::nopro reset from > to", oss.str());
    }

  if ( vm.count("mode") )
    {
      if (    simulationMode == 0        // meaning less than 'xeona::e_hollowCall'
           || simulationMode  > xeona::e_fullRun )
        {
          logger->repx(logga::dbug, "simulation mode not supported", simulationMode);
          std::ostringstream oss;
          oss << "'mode' " << simulationMode << " is not supported";
          throw std::domain_error(oss.str());      // see Lischner (2003 p658)
        }
      simKind = static_cast<xeona::SimKind>(simulationMode);
    }

  if ( vm.count("nodata") )
    {
      // the following removes the need for conditionals when logging
      const logga::Rank wasRank = logga::rankNoData;    // logga::dbug
      logga::rankNoData         = logga::warn;
      std::ostringstream oss;
      oss << wasRank << " > " << logga::rankNoData;
      logger->repx(logga::dbug, "logga::rankNoData reset from > to", oss.str());
    }

  if ( vm.count("jumpy") )
    {
      // the following removes the need for conditionals when logging
      const logga::Rank wasRank = logga::rankJumpy;     // logga::dbug
      logga::rankJumpy          = logga::warn;
      std::ostringstream oss;
      oss << wasRank << " > " << logga::rankJumpy;
      logger->repx(logga::dbug, "logga::rankJumpy reset from > to", oss.str());
    }

  if ( vm.count("pepper") )
    {
      const bool was = xeona::pepper;              // set 'false' in 'common.cc'
      if ( xeona::pepper ) xeona::pepper = false;  // toggle under '--pepper'
```

```
      else                xeona::pepper = true;
      std::ostringstream oss;
      oss << std::boolalpha;                        // use labels
      oss << was << " > " << xeona::pepper;
      logger->repx(logga::dbug, "xeona::pepper reset from > to", oss.str());
    }

  if ( vm.count("report") )
    {
      const unsigned clip                       // clip value
        = static_cast<unsigned>(logga::adhc) + 1;
      if ( reportLevel > clip )
        {
          reportLevel = clip;
          logger->repx(logga::dbug, "clipping --report arg back to", clip);
        }
      if ( reportLevel == clip )
        {
          logga::Logger::disableConsoleTruncation();
          logger->repx(logga::dbug, "console log truncation disabled", "");
          reportLevel = clip - 1;
        }
      logger->repx(logga::dbug,
                   "will set report level to value",
                   reportLevel);
      logger->setReportLevel(reportLevel);    // active call
    }
  else                                        // otherwise reset to 'common' default
    {
      logger->repx(logga::dbug,
                   "will reset report level to default",
                   xeona::reportLevelDefault);
      logger->setReportLevel(xeona::reportLevelDefault);
    }

  if ( vm.count("tidy") )                // alias to --mode 3 --report 3
    {
      logger->setReportLevel(3);         // active call
      simKind = xeona::e_parseModel;     // active setting
      logger->repx(logga::info, "option --tidy actioned", "");
    }

  if ( vm.count("watch") && ! logTrigger.empty() )
    {
      if ( boost::find_token(logTrigger, boost::is_space()) )
        {
          std::ostringstream oss;
          oss << "'watch' argument cannot contain whitespace: "
              << logTrigger;
          throw std::domain_error(oss.str()); // see Lischner (2003 p658)
        }
      else
        {
          logger->repx(logga::info, "watch option set", logTrigger);
          logger->setTrigger(logTrigger);     // enable specific reporting
        }
    }

  if ( vm.count("yeek") )
    {
      const unsigned was = xeona::yeek;             // as set in 'common.cc'
      xeona::yeek        = yeekValue;               // update
      std::ostringstream oss;
      oss << was << " > " << xeona::yeek;
      logger->repx(logga::dbug, "xeona::yeek reset from > to", oss.str());

      // interpretation reporting
      std::string msg = xeona::yeekInterpret(yeekValue);
      if ( msg.empty() )
        {
          msg = "(not in database)";
          logger->repx(logga::warn, "yeek value not in database", yeekValue);
        }
      std::ostringstream put;
      put << "  yeek value : " << yeekValue << " = " << msg << "\n";
      logger->putx(logga::xtra, put);
    }

  if ( vm.count("zero") )
    {
      const bool was = xeona::zero;                 // set 'true' in 'common.cc'
```

```
        if ( xeona::zero ) xeona::zero = false;      // toggle under '--zero'
        else               xeona::zero = true;
        std::ostringstream oss;
        oss << std::boolalpha;                       // use labels
        oss << was << " > " << xeona::zero;
        logger->repx(logga::dbug, "xeona::zero reset from > to", oss.str());
      }

  } // command-line 'try' block

// ---------------------------------
//  command-line 'catch' blocks
// ---------------------------------

// CAUTION: 'stdout' and not 'stderr' used for direct reporting
// as file descriptor 2 may well be redirected to '/dev/null'

catch( const std::exception& e )
  {
    std::clog << std::flush;
    std::cout
      << appInfo.getProgramName()
      << ": incorrect usage (try --help): "
      << e.what()
      << std::endl;
    logger->disableBeeping();              // no need to beep
    return xeona::ret_usage;               // return
  }
catch( ... )
  {
    std::clog << std::flush;
    std::cout
      << appInfo.getProgramName()
      << ": boost::program_options exception of unknown type"
      << std::endl;
  }

// ---------------------------------
//  display splash and info screens
// ---------------------------------

logger->repx(logga::dbug, "splash and info screen calls", "stdout");

std::clog << std::flush;
std::cout << appInfo.showSplash();         // first six lines, including blanks
std::cout << appInfo.showInfo(simKind, beepMode, exitTripLevel);
std::cout << std::flush;

// =================================
//  simulation call 'try' block
// =================================

// NOTE: the 'xeona::' variables are set in 'common.cc'

try
  {
    std::string modelName = "";            // set via '--inbuilt' or '--guard' or null

    // ---------------------------------
    //  generate inbuilt model
    // ---------------------------------

    if ( flag_generateInbuiltFile )        // set by '--inbuilt'
      {
        // unit 'inbuilt'
        modelName = xeona::createModelName(cmdLineModelName);
        logger->repx(logga::info, "about to create test model", modelName);
        logger->repx(logga::info, "inbuilt steps set to", inbuiltSteps);
        const bool ret = xeona::dumpToFile(modelName, inbuiltSteps);
        if ( ret )
          logger->repx(logga::dbug, "write to file succeeded", ret);
        else
          logger->repx(logga::warn, "write to file failed", ret);
      }

    // ---------------------------------
    //  regenerate external test model
    // ---------------------------------

    else if ( flag_refreshModelIfPossible )          // set by '--guard'
      {
```

```
    // obtain stub
    std::string stub(cmdLineModelName);     // block local copy
    if ( stub.empty() )
      {
        stub = xeona::modelStubDefault;
      }
    if ( boost::ends_with(stub, xeona::modelExt) )         // trim ".xem"
      {
        boost::erase_tail(stub, xeona::modelExt.length());
      }
    if ( boost::ends_with(stub, xeona::modelGuardTag) )    // trim ".guard"
      {
        boost::erase_tail(stub, xeona::modelGuardTag.length());
      }

    // build file names
    std::string modelname = stub;
    modelname            += xeona::modelExt;          // add ".xem"
    std::string guardname = stub;
    guardname            += xeona::modelGuardTag;     // add ".guard"
    guardname            += xeona::modelExt;          // add ".xem"

    // create paths
    boost::filesystem::path model = boost::filesystem::absolute(modelname);
    boost::filesystem::path guard = boost::filesystem::absolute(guardname);

    // copy over file
    if ( exists(guard) )
      {
        logger->repx(logga::dbug, "reinstating model from", guard.filename());
        boost::filesystem::remove(model);  // copy_file will not overwrite
        boost::filesystem::copy_file(guard, model);
      }
    else
      {
        logger->repx(logga::warn, "guard file not found", guard.filename());
        logger->repx(logga::info, "not possible to reinstate model", "");
        throw xeona::file_not_found(guard.filename().string(), "guard file");
      }

    modelName = modelname;
  }

// ---------------------------------
//  straight name
// ---------------------------------

else
  {
    // test for presence of ".guard"
    std::string stub(cmdLineModelName);                    // block local copy
    if ( boost::ends_with(stub, xeona::modelExt) )         // trim ".xem"
      {
        boost::erase_tail(stub, xeona::modelExt.length());
      }
    if ( boost::ends_with(stub, xeona::modelGuardTag) )    // trim ".guard"
      {
        throw xeona::cannot_run_guard(cmdLineModelName);   // choke
      }
    modelName = cmdLineModelName;
  }

// ---------------------------------
//  run simulation, see unit 'simcall'
// ---------------------------------

const std::string value = '"' + modelName + '"';
logger->repx(logga::dbug, "about to call simulate using", value);

sret = simulate(modelName,             // from command-line or default
                simKind);              // from '--mode' option or default

// ---------------------------------
//  reprocess 'sret'
// ---------------------------------

int contextBasedReturn = -1;                                  // nonsensical value
if ( xeona::releaseStatus == true ) contextBasedReturn = xeona::ret_test_code_used;
else                                contextBasedReturn = 0;  // ease of development

switch ( sret )                        // simulation call return
```

```
      {
case xeona::e_success:          returnStatus = xeona::ret_success;          break;
case xeona::e_modelFileFault:   returnStatus = xeona::ret_model_file_fault; break;
case xeona::e_errantSimulation: returnStatus = xeona::ret_errant_simulation; break;
case xeona::e_infeasibility:    returnStatus = xeona::ret_infeasibility;    break;
case xeona::e_testCodeUsed:     returnStatus = contextBasedReturn;          break;
case xeona::e_other:            returnStatus = xeona::ret_other;            break;
case xeona::e_statusNotKnown:   returnStatus = xeona::ret_status_not_known; break;
default: std::clog << "** coding error 02 in source file " << __FILE__ << std::endl;
      } // 'switch' block

  } // 'try' block

// ---------------------------------
//  simulation call 'catch' blocks
// ---------------------------------

// CAUTION: 'stdout' and not 'stderr' used for direct reporting
// as file descriptor 2 may well be redirected to '/dev/null'

catch( const xeona::kill_on_log& x )
  {
    logger->repx(logga::info, "xeona::kill_on_log caught", "");
    logger->flush();
    returnStatus = x.code();
  }
catch( const xeona::file_not_found& x )
  {
    logger->repx(logga::info, "xeona::file_not_found caught", "");
    logger->flush();
    returnStatus = x.code();
  }
catch( const xeona::exception& x )
  {
    logger->repx(logga::info, "polymorphic xeona::exception caught", "");
    std::ostringstream put;
    put << x.expl();
    logger->putx(logga::info, put);
    logger->flush();
    std::cout
      << "\n"
      << "** " << x.tell() << " caught, execution abandoned"            << "\n"
      << std::flush;
    logger->addSmartBlank();
    returnStatus = x.code();
  }
catch( const xeona::bad_assign_cast& a )   // inherits direct from 'std::exception'
  {
    logger->repx(logga::info, "xeona::bad_assign_cast caught", "");
    logger->flush();
    std::cout
      << "\n"
      << "** xeona::bad_assign_cast: " << a.what()                     << "\n"
      << "   for more information, rerun model with maximum reporting"  << "\n"
      << std::flush;
    logger->addSmartBlank();
    returnStatus = xeona::exit_bad_assign_cast;
  }
catch ( const boost::bad_weak_ptr& p )     // place before 'std::exception'
  {
    logger->repx(logga::info, "boost::bad_weak_ptr caught", "");
    logger->flush();
    std::cout
      << "\n"
      << "** boost::bad_weak_ptr: " << p.what()                        << "\n"
      << "   for more information, rerun model with maximum reporting"  << "\n"
      << std::flush;
    logger->addSmartBlank();
    returnStatus = xeona::exit_boost_exception;
  }
catch(const std::out_of_range& e)
  {
    logger->repx(logga::info, "std::out_of_range caught", "");
    logger->flush();
    std::cout
      << "\n"
      << "** std::out_of_range: " << e.what()                                 << "\n"
      << "   possible cause: inappropriate 'at' call by 'std::vector' object" << "\n"
      << "   for more information, rerun model with maximum reporting"        << "\n"
      << std::flush;
    logger->addSmartBlank();
```

```
      returnStatus = xeona::exit_std_out_of_range;
    }
  catch(const std::logic_error& e)
    {
      logger->repx(logga::info, "std::logic_error caught", "");
      logger->flush();
      std::cout
        << "\n"
        << "** std::logic_error: " << e.what()                                    << "\n"
        << "   for more information, rerun model with maximum reporting"         << "\n"
        << std::flush;
      logger->addSmartBlank();
      returnStatus = xeona::exit_std_logic_error;
    }
  catch(const std::bad_alloc& m)
    {
      logger->repx(logga::info, "std::bad_alloc caught", "");
      logger->flush();
      std::cout
        << "\n"
        << "** std::bad_alloc: " << m.what()                                      << "\n"
        << "   cause: insufficient heap (mid-simulation) memory"                  << "\n"
        << "   for more information, rerun model using a memory profiler"         << "\n"
        << std::flush;
      logger->addSmartBlank();
      returnStatus = xeona::exit_std_bad_alloc;
    }
  catch(const std::exception& e)
    {
      logger->repx(logga::info, "std::exception caught", "");
      logger->flush();
      std::cout
        << "\n"
        << "** std::exception: " << e.what()                                 << "\n"
        << "   for more information, rerun model with maximum reporting"     << "\n"
        << std::flush;
      logger->addSmartBlank();
      returnStatus = xeona::exit_std_exception;
    }
  catch (...)
    {
      logger->repx(logga::warn, "unknown exception caught", "");  // note 'logga::warn'
      logger->flush();
      std::cout
        << "\n"
        << "** unknown (...) exception"                                      << "\n"
        << "   for more information, rerun model with maximum reporting"     << "\n"
        << std::flush;
      logger->addSmartBlank();
      returnStatus = xeona::exit_unknown_exception;
    }

  // ================================
  //  conclusion
  // ================================

  // --------------------------------
  //  reprocess return if --exittrip
  // --------------------------------

  // CAUTION: program execution may not get here due to 'return'
  // statements in the above code, however 'returnStatus' should
  // aways be overwritten -- that is also tested here

  if ( returnStatus == xeona::ret_success )
    {
      for ( unsigned level = 0;             // type because 'exitTripLevel' is unsigned
            level <= exitTripLevel;
            level++ )                       // count from zero to 'exitTripLevel'
        {
          logger->repx(logga::adhc, "current exit trip level", level);
          switch ( level )
            {
            case logga::yeek:
              if ( logger->getYeekAllCount() > 0 ) returnStatus = logga::yeek;
              break;
            case logga::kill:
              if ( logger->getKillAllCount() > 0 ) returnStatus = logga::kill;
              break;
            case logga::warn:
              if ( logger->getWarnAllCount() > 0 ) returnStatus = logga::warn;
```

```
              break;
            case logga::info:
              if ( logger->getInfoAllCount() > 0 ) returnStatus = logga::info;
              break;
            case logga::dbug:
              if ( logger->getDbugAllCount() > 0 ) returnStatus = logga::dbug;
              break;
            case logga::xtra:
              if ( logger->getXtraAllCount() > 0 ) returnStatus = logga::xtra;
              break;
            case logga::adhc:
              if ( logger->getAdhcAllCount() > 0 ) returnStatus = logga::adhc;
              break;
            default:
              std::clog << "** coding error 03 in source file " << __FILE__ << std::endl;
              break;
            }
          if ( returnStatus > 0 ) break;     // no need to go further
        }
    }
  else if ( returnStatus == xeona::ret_not_overwritten )
    {
      logger->repx(logga::warn, "main function return not overwritten", returnStatus);
    }

  // ---------------------------------
  //  pass return status to logger
  // ---------------------------------

  // the reason for this call is that several screens-full of
  // output can follow the final report splash screen -- this
  // information reported when the Logger instance finally
  // destructs.

  if ( returnStatus == 0 )
    {
      logger->updateReturnStatus(returnStatus, "success");  // short form message
    }
  else
    {
      std::string interpretation;            // fill by reference
      ExitStatus exitstatus;                 // accept default not-present message
      exitstatus(returnStatus, interpretation);
      logger->updateReturnStatus(returnStatus, interpretation);
    }

  // ---------------------------------
  //  display final report
  // ---------------------------------

  std::clog << std::flush;
  std::cout << appInfo.showFinal(sret, returnStatus);

  std::cout << std::flush;
  std::clog << std::flush;

  // ---------------------------------
  //  exit with an appropriate code
  // ---------------------------------

  logger->repx(logga::dbug, "end of main", "~~~~~~~~~~");
  return returnStatus;

} // main

//  end of file
```

```
#  XEONA ENERGY SYSTEMS MODELING ENVIRONMENT

#  file-purpose     : project makefile (requires clean svn, unlike 'mach')
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Wed 11-Apr-2007 12:10 UTC
#  file-status      : working
#  file-keywords    : xeona

#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeona1/makefile $

#  ------------------------------------------------------------
#  fastdep
#  ------------------------------------------------------------
#
#  This makefile utilizes the 'fastdep' utility (v0.16)
#  by Bart Vanhauwaert to generate file dependencies.
#  The equivalent scripted code using g++ -M and sed
#  calls is rather convoluted.  (If you wish to avoid
#  third party utilities, refer particularly to Stephens
#  etal (2006 p75-76) and to the GNU make info
#  documentation "4.14 Generating Prerequisites
#  Automatically" for sample code).
#
#  From 04-Feb-2010, an earlier fastdep file (as named
#  by 'fastdep_file') will be recycled if the fastdep
#  utility cannot be located.  This strategy assumes
#  that all '#include' statements remain unchanged and
#  should be considered a temporary measure.  Indeed, it
#  is highly recommended that fastdep be installed and
#  used.
#
#  If fastdep proves an impediment, it may be better to
#  replicate fastdep as a local script using g++ -M*
#  calls.)
#
#  ------------------------------------------------------------
#  Overview
#  ------------------------------------------------------------
#
#  In terms of references, Welsh etal (2003, pp437--447)
#  give a good review of make.  In addition, Stephens etal
#  (2006, pp 64-82) provide an excellent coverage (with
#  examples) of advanced topics like non-static rules and
#  command-line customization.  Alternatively, basic
#  multi-file makefiles are described by Oualline (1995,
#  pp 420-424).
#
#  The GNU info documentation is very comprehensive.  See,
#  in particular, "10 Using Implicit Rules"
#
#  ------------------------------------------------------------
#  References for make
#  ------------------------------------------------------------
#
#  Standard documentation for GNU make
#
#  Oualline, Steve.  1995.  Practical C++ programming
#    -- First edition.  O'Reilly and Associates,
#    Sebastopol, USA.  ISBN 1-56592-139-9.
#
#  Stephens, D Ryan, Christopher Diggins, Jonathan
#    Turkanis, and Jeff Cogswell.  2006.  C++ cookbook.
#    O'Reilly and Associates, Sebastopol, USA.  ISBN
#    0-596-00761-2.
#
#  Welsh, Matt, Matthias Kalle Dalheimer, Terry Dawson,
#    Lar Laufman.  2003.  Running Linux -- Fourth edition.
#    O'Reilly and Associates, Sebastopol, USA.  ISBN
#    0-596-00272-6.  [see above]
#
#  ------------------------------------------------------------
#  GNU make options
#  ------------------------------------------------------------
#
#    --jobs=4
#        better utilize 4 core processors, results in a
#        more than 200% speed up on the Intel Core i5
#
#        certain caveats apply so omit on release builds
#
```

```
#  ------------------------------------------------------
#  GNU g++ 4 compiler options
#  ------------------------------------------------------
#
#    -###
#        like -v except the commands are not executed and
#        all command arguments are quoted, will probably
#        return a fail exit status (perhaps 2) if the linker
#        is invoked
#
#    -v
#        print (on standard error output) the commands
#        executed to run the stages of compilation -- also
#        print the version number of the compiler driver
#        program and of the preprocessor and the compiler
#        proper
#
#    -g
#        produce debugging information in the operating
#        system's native format (stabs, COFF, XCOFF, or
#        DWARF 2), also okay for GDB
#
#    -ggdb
#        produce debugging information for use by GDB -- this
#        means to use the most expressive format available
#
#    -fno-inline
#        ignore the "inline" keyword (effective only under
#        optimization)
#
#    -O3
#        maximum level of optimization for speed
#
#    -Wall
#        enable all warnings about questionable
#        constructs, see also -Wextra for constructs not
#        often regarded as questionable
#
#    -pedantic [-Wno-long-long]
#        issue all the warnings demanded by strict ISO C
#        and ISO C++ [Boost 1.35 and better requires long long
#        errors to be suppressed]
#
#    -Weffc++
#        warn about bad style as defined in Scott Meyers'
#        Effective C++ book (may also need to
#        | grep --invert-match "boost::" and so forth)
#
#  ------------------------------------------------------
#  Third-party libraries
#  ------------------------------------------------------
#
#  All have installed as pre-built ubuntu packages, but some
#  may later have been locally built to yield a newer release.
#
#    libglpk                   : Andrew Makhorin's MILP solver
#    lboost_filesystem         : Boost filesystem operations
#    lboost_regex              : Boost regular expression support
#    libboost_date_time        : Boost date and time support
#    libboost_program_options  : Boost command-line and config file parser
#
#  ------------------------------------------------------

#---------------------------------
# internal variables
#---------------------------------

# NOTE: naming convention: as recommended in the GNU
# documentation for make:
#
#   internal variables : lowercase
#   external variables : uppercase

# CAUTION: best not to add trailing comments to
# definition statements

# binary         : name of binary
# binary_opts    : options to be passed to binary at run-time (no double quotes)
# common         : stem for common header
# install_dir    : subdirectory to install program
# fastdep_file   : file generated by fastdep
```

```
# clean_exts     : list of file extensions to remove
# max_depth      : 'find -maxdepth' level, 1 = base director only
# script         : used only for console messages

  binary       = xeona.make
  binary_opts  = --beep 1 --report 6 --inbuilt 6

  common       = common
  install_dir  = ./bin
  fastdep_file = dependsfile

  clean_exts   = .o
  max_depth    = 2

  script       = $(firstword $(MAKEFILE_LIST))

#---------------------------------
# external variables
#---------------------------------

# Usage notes:
#
#  * simply comment out unwanted definitions (best to
#    use a double ##)
#
#  * or redefine on the command-line, for instance to
#    cancel -D_XDEBUG or -D_XSVNREV, use respectively
#
#      $ make CPPFLAG_DEBUG=  [all]
#      $ make CPPFLAG_SVNREV= [all]
#
#  * to combine command-line and makefile settings, use
#    the 'override' directive and the += operator (see
#    GNU manual, section 6.7)
#
# Note: implicit variables from make are ALWAYS
# uppercase, so too should any user-declared variables
# that might need to be overwritten on the command-line
# (although this second guideline was not adhered to
# here in respect of 'sources'!)

# The application will record the prevailing subversion
# revision if the macro '_XSVNREV' is set to a valid
# integer.  The following code does this iff the
# revision control is up-to-date.  Note that only
# 'common.cc' need be rebuilt -- and that always
# happens due to the 'touch' command later.  Note also
# the word "shell" in the following statement

# The code stanza below sets RES to zero if the
# 'svnversion' call returns the phrase "exported".
# This phrase means that the makefile no longer resides
# within a directory under subversion version control.
# For more information on syntax, see info make.

  RES = $(shell svnversion .. 2>/dev/null)
  ifndef $(RES)          # svnversion not present
    RES = 0
  endif
  ifeq ($(RES),exported) # present but outside repository
    RES = 0
  endif
  CPPFLAG_SVNREV = -D_XSVNREV=$(RES)

# CAUTION: do not use trailing comments in the above
# assignments!

# To bypass the above protective code for testing
# purposes, use the following macro:
#
#   $ make CPPFLAG_SVNREV="-D_XSVNREV=0"

# The debug flag is also independently defined

  CPPFLAG_DEBUG  = -D_XDEBUG

# CXX        : compiler call          defaults to g++
# CXXFLAGS   : compiler flags         defaults to nil
# CPPFLAGS   : preprocessor flags     defaults to nil
# LDFLAGS    : linker flags           defaults to nil
# LDLIBS     : link-time libraries    defaults to nil
```

```
# Note GCC linker option -static for static linking,
# which can be enacted via LDFLAGS.

# The following debug build options are hard-coded
# '_XTCOLS' on 'hinau' now 166, was 142 for 'sojus'

  CXX         = g++

  CXXFLAGS   += -Wall
  CXXFLAGS   += -ggdb
  CXXFLAGS   += -O0
  CXXFLAGS   += -fno-inline

  CPPFLAGS   += $(CPPFLAG_DEBUG)
  CPPFLAGS   += $(CPPFLAG_SVNREV)
  CPPFLAGS   += -D_XTCOLS=166

  LDFLAGS    +=

# A cross-compiling trial using mingwin on Linux (by
# Peter Salhmann) normally deploys the following make
# two options:
#
#     CROSS="i686-pc-mingw32-"
#     -I "/opt/mingw/usr/i686-pc-mingw32/include"
#
# Given the above, the following code ensures that the
# cross-compiled binary will NOT be linked with shared
# libraries.  And also that a different binary naming
# convention will be used.

  ifdef CROSS                 # cross-compile
    LDFLAGS += -static
  endif

  ifdef CROSS                 # cross-compile
    ifeq ($(RES),0)
      binary = xeona.exe
    else
      binary = xeona$(RES).exe
    endif
  endif

# The GNU MP bignum library (libgmp) is required
# because the GLPK library (libglpk) was configured
# using --with-gmp.  If needed, bignum support can be
# omitted.

  LDLIBS     += -lboost_date_time
  LDLIBS     += -lboost_filesystem
  LDLIBS     += -lboost_program_options
  LDLIBS     += -lboost_regex
  LDLIBS     += -lboost_system
  LDLIBS     += -lglpk
# LDLIBS     += -lgmp

# The POSIX threads library (lpthread) needs to be
# added to LDLIBS if LDFLAG contains '-static'.  The
# '-static' option prevents the compiler linking with
# shared libraries, in which case 'ldd' then reports
# "not a dynamic executable".  The following stanza
# simply relieves the caller of the need to modify
# LDLIBS under these circumstances.  Note that 'xeona'
# itself is NOT multi-threaded.

  ifneq (,$(findstring -static,$(LDFLAGS)))
    LDLIBS   += -lpthread
  endif

# the Boost libraries link to release 1.35 and better --
# to backtrack to release 1.34.1, remove the trailing
# "-mt" and set CPATH="/usr/local/include/boost-1_34_1"

# explicit linking with 'libgmp' was required from GLPK
# 4.33, but only when GLPK is configured with
# '--with-gmp'

#---------------------------------
# source list
#---------------------------------
```

```
# CAUTION: 'make' will stop at, rather than jump, the
# first commented out line, also leave space between
# filename and slash or 'make' will crunch the entries
# together

sources = \
        ./$(common).cc \
        a/appinfo.cc \
        a/exapp.cc \
        a/exbase.cc \
        a/exent.cc \
        a/exitstat.cc \
        a/floatstat.cc \
        a/helpers.cc \
        a/logger.cc \
        a/recorder.cc \
        a/xedocs.cc \
        a/xemopt.cc \
        a/yeek.cc \
        b/actor.cc \
        b/asop.cc \
        b/asop01.cc \
        b/asop02.cc \
        b/asop03.cc \
        b/auxs01.cc \
        b/bandtaf.cc \
        b/block.cc \
        b/builtins.cc \
        b/commods.cc \
        b/commods01.cc \
        b/costreg.cc \
        b/domcon.cc \
        b/entity.cc \
        b/gate.cc \
        b/gate01.cc \
        b/junc.cc \
        b/junc01.cc \
        b/junc02.cc \
        b/lmpbid.cc \
        b/node.cc \
        b/node01.cc \
        b/node02.cc \
        b/optctl.cc \
        b/optgate.cc \
        b/optjunc.cc \
        b/optnode.cc \
        b/optops.cc \
        b/optprob.cc \
        b/overseer.cc \
        b/propdata.cc \
        b/register.cc \
        b/teas.cc \
        b/teas01.cc \
        b/teas02.cc \
        b/teas03.cc \
        b/teas04.cc \
        b/teasdev.cc \
        b/tests.cc \
        b/tictoc.cc \
        c/conex.cc \
        c/costs.cc \
        c/datio.cc \
        c/extunits.cc \
        c/factory.cc \
        c/files.cc \
        c/fincalc.cc \
        c/ghouse.cc \
        c/inbuilt.cc \
        c/label.cc \
        c/linklog.cc \
        c/recset.cc \
        c/si3units.cc \
        c/simcall.cc \
        c/stats.cc \
        c/tsops.cc \
        c/util1.cc \
        c/util2.cc \
        c/util3.cc \
        c/xemgen.cc \
        c/xeona_ptr.cc \
```

```
        d/glpkviz.cc \
        d/siglp.cc \
        e/context.cc \
        e/cxamb01.cc \
        e/cxamb02.cc \
        e/cxecon01.cc \
        e/cxpol01.cc \
        f/cta.cc \
        f/gatesreg.cc \
        f/ospinfo.cc \
        f/trav.cc \
        ./main.cc \

#---------------------------------
# unaccompanied headers list
#---------------------------------

unheaders = \
        a/logger_fwd.h \
        a/license.h \
        c/smart_ptr.h \
        f/ospmodes.h \

#---------------------------------
# phony and suffix lists
#---------------------------------

# the phony target prevents similarly named files from
# blocking the execution of makefile targets

.PHONY: .compile_start .confirm_fastdep .depends .cross_echo .sources_echo
.PHONY: all ccs check clean deps h help inst ls rmbin tidy uninst

# reset suffix list

.SUFFIXES:
.SUFFIXES: .cc .o

#---------------------------------
# check for fastdep (used later)
#---------------------------------

.confirm_fastdep:
        @which fastdep >/dev/null \
          || echo "$(script): WARN: fastdep utility not present"

#---------------------------------
# build commands
#---------------------------------

# the touch common.cc call ensures that the xeona::DBUG
# variable, in turn set by the _XDEBUG preprocessor
# macro, will propogate through the app at link-time

objects = $(patsubst %.cc, %.o, $(sources))  # simple file extension substitution

# overarching call -- note use of reporting targets for timely information

all: .confirm_fastdep .cross_echo .sources_echo .depends .compile_start $(binary)
        @echo "$(script): completion of makefile target: all"

.cross_echo:
  ifdef CROSS
        @echo "$(script): CROSS: $(CROSS)"
  endif

.sources_echo:
        @echo "$(script): sources: $(sources)"

.compile_start:
        @echo "$(script): touching $(common).cc"; \
          touch $(common).cc
        @echo "$(script): building objects" # "(perhaps)"

# linking call

$(binary): $(objects)
        @echo "$(script): linking binary: $(binary)"
        @sleep 1 # to ensure binary is listed oldest on ls -t (sort by modification time)
        $(CXX) $(LDFLAGS) -o $@ $^ $(LDLIBS)
```

```
# compilation calls -- often implicit but hardcoded here for safety's sake

%.o: %.cc
	$(CXX) $(CXXFLAGS) $(CPPFLAGS) -c -o $@  $<

#---------------------------------
# dependency list generation
#---------------------------------

# CAUTION: "Put the code to generate dependencies at
# end of your makefile" (Stephens etal, 2006, p75)

# the following code will recycle an earlier fastdep
# header dependencies file if fastdep itself cannot be
# located -- see the fastdep section (at the top of
# this file) for more information

# CAUTION: note the $$ when using bash command substitution

msg1 = fastdep header dependencies

.depends:
	@if test $$( which fastdep ); then \
	  echo "$(script): executing fastdep"; fastdep $(sources) > $(fastdep_file); \
	  else echo "$(script): WARN: unable to freshen $(msg1)"; fi

	@test -s $(fastdep_file) \
	  || echo "$(script): ERROR: substantive $(msg1) file not found: $(fastdep_file)"

-include $(fastdep_file)  # leading - suppresses 'no such file' warnings under multi-pass

#---------------------------------
# utility calls
#---------------------------------

inst:
	@mkdir -p $(install_dir)
	@cp -p $(binary) $(install_dir)
	@echo "$(script): install complete"
	@ls -lF $(install_dir)/$(binary)

uninst:
	@rm --force --recursive $(install_dir)
	@echo "$(script): uninstall complete"

check:
	@echo "$(script): attempting to execute binary: $(binary) $(binary_opts)"
	@if test -x $(binary); then \
	  echo; ./$(binary) $(binary_opts); echo "\n$(script): binary exit: $$?"; \
	  else echo "$(script): $(binary) missing or not executable (test -x failed)"; fi

deps: .confirm_fastdep .depends
	@echo "$(script): using fastdep file: $(fastdep_file)"
	@echo "$(script): using sources: $(sources)"
	@echo
	@cat $(fastdep_file)
	@echo

ls:
	@echo -n "$(script): name: "
	@ls [Mm]akefile*

ccs:
	@echo $(sources) | awk 'BEGIN { RS = " "; print "" } { print $0 }'

hs:
	@echo $(unheaders) | awk 'BEGIN { RS = " "; print "" } { print $0 }'

#---------------------------------
# cleanup calls
#---------------------------------

tidy:
	@echo "$(script): removing empty files to depth $(max_depth)"; \
	  find . -maxdepth $(max_depth) -type f -empty -delete
	@for ext in $(clean_exts); \
	  do echo "$(script): removing *$$ext to depth $(max_depth)" ; \
	    find . -maxdepth $(max_depth) -type f -name "*$$ext" -delete ; \
	  done

rmbin:
```

```
        @echo "$(script): removing $(binary)"; \
          rm --force $(binary)

clean: tidy rmbin

#---------------------------------
# help message
#---------------------------------

h: help   # short-cut to help

help:
        @ echo
        @ echo "  action:"
        @ echo "     [all]       rebuild as required"
        @ echo "     deps        update and cat fastdep file"
        @ echo "     tidy        remove created files but leave binary"
        @ echo "     rmbin       remove binary"
        @ echo "     clean       tidy + rmbin"
        @ echo "     inst        install program in $(install_dir)/"
        @ echo "     uninst      prune branch $(install_dir)/"
        @ echo "     check       attempt to run program"
        @ echo "  info:"
        @ echo "     help|h      this message"
        @ echo "     ls          list makefiles"
        @ echo "     ccs         list source files"
        @ echo "     hs          list unaccompanied header files"
        @ echo "  internal variables:"
        @ echo "     fastdep file (fastdep_file)   : $(fastdep_file)"
        @ echo "     binary name (binary)          : $(binary)"
        @ echo "     binary options (binary_opts)  : $(binary_opts)"
        @ echo "  external variables (as currently overwritten):"
        @ echo "     compiler call (CXX)           : $(CXX)"
        @ echo "     compiler flags (CXXFLAGS)     : $(CXXFLAGS)"
        @ echo "     preprocessor flags (CPPFLAGS) : $(CPPFLAGS)"
        @ echo "     linker flags (LDFLAGS)        : $(LDFLAGS)"
        @ echo "     link-time libraries (LDLIBS)  : $(LDLIBS)"
        @ echo "  problems with $(common).cc?"
        @ echo "     first, ensure that the revision control is up-to-date"
        @ echo "     then, try setting CPPFLAG_SVNREV to nil from the command-line"
        @ echo

#  ------------------------------------------------------------
#  GNU make database (extract)
#  ------------------------------------------------------------
#
#  special meaning    : LDFLAGS : extra flags to give linker
#  no special meaning : TARGET_ARCH LOADLIBES LDLIBS
#  not set by default : CXXFLAGS CPPFLAGS LDFLAGS TARGET_ARCH LOADLIBES LDLIBS
#
#  # Variables
#
#  # default
#  CXX = g++
#
#  # default
#  LINK.cc    = $(CXX) $(CXXFLAGS) $(CPPFLAGS) $(LDFLAGS) $(TARGET_ARCH)
#
#  # default
#  COMPILE.cc = $(CXX) $(CXXFLAGS) $(CPPFLAGS) $(TARGET_ARCH) -c
#
#  # default
#  LINK.o = $(CC) $(LDFLAGS) $(TARGET_ARCH)
#
#  # default
#  OUTPUT_OPTION = -o $@
#
#  # Implicit Rules
#
#  %: %.cc
#  #  commands to execute (built-in):
#          $(LINK.cc) $^ $(LOADLIBES) $(LDLIBS) -o $@
#
#  %.o: %.cc
#  #  commands to execute (built-in):
#          $(COMPILE.cc) $(OUTPUT_OPTION) $<
#
#  ------------------------------------------------------------

#---------------------------------
# junk code
```

```
#---------------------------------

# @fastdep --extraremakedep=$(script) --remakedeptarget=.depends $(sources) \
#   > $(fastdep_file)

# define a special CPPFLAG when building on 'sojus',
# the principal development host

#   MYSELF = $(shell hostname --long)
#   ifeq ($(MYSELF),sojus.fb10.tu-berlin.de)
#     CPPFLAGS += -D_XWEFF
#   endif

#  end of file
```

```
#! /bin/bash

#  file-purpose     : extract entity documentation from 'xeona' headers
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Fri 08-Aug-2008 08:18 UTC
#  file-status      : working
#  file-keywords    : xeona

#  $Revision: 3496 $
#  $Date: 2009-10-09 10:39:32 +0200 (Fri, 09 Oct 2009) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/scripts/xedoc $

# ---------------------------------
#  representative entry
# ---------------------------------

#  //  ==== XEDOC ================================================
#  //
#  //  entity.actor-0
#  //
#  //      class                                   > Actor
#  //
#  //      builtin-remark s                        <
#  //
#  //      this class is highly incomplete and should not be used
#  //
#  //  ============================================================

#  CAUTION: this script is designed to be reasonably
#  tolerant to typos -- however:
#
#      exactly four leading '=' must be used
#      'XEDOC' must be spelled and cased exactly as shown
#      left padding (to the comment string) must be zero, one, or two spaces
#
#  default "relative to comment string" 'tab-stop-list' is ( 04 45 47)

# ---------------------------------
#  settings
# ---------------------------------

MAINCC="main.cc"                        # name of main file, used to confirm base directory

# ---------------------------------
#  preamble
# ---------------------------------

SCRIPT=$(basename "$0")

E_SUCCESS=0
E_FAILURE=1
E_USAGE=2
E_NO_MAIN=10
E_NO_HEADERS_IDENTIFIED=11

# ---------------------------------
#  display_help()
# ---------------------------------

function display_help
{

    echo
    echo "         usage: $SCRIPT              list the model requirements for entities"
    echo "                $SCRIPT  --include   list as above in special xeona #include format"
    echo "                $SCRIPT  --show      display a skeleton entry"
    echo "                $SCRIPT  --help      display this message and exit"
    echo "       purpose: scans appropriate headers and extracts entity requirements directly from the ⬋
code"
    echo "      activity: this script is entirely passive"
    echo "         notes: - test entities are not suitably documented and therefore not recovered"
    echo "                - informational reporting uses 'stderr' to facilitate dumping"
    echo "                - see also 'emacs f10 >' to reindent XEM files"
    echo "     hardcodes: subdirectory: $SDIR"
    test $(which gawk) || echo "       WARNING: 'gawk' utility not found on your system"
    echo
}

# ---------------------------------
#  print_example()
```

```
# ---------------------------------

# here-documents:
#
# the escaped EOF means no "parameter substitution,
# command substitution, and arithmetic expansion"
# (bash manpage)
#
# there must be no trailing chars after the final "EOF"

function print_example
{
cat << \EOF                             # the "\EOF" is escaped

//  ==== XEDOC ==========================================
//
//  entity.class-0
//
//      class                                   > Class
//
//        some comment
//
//      builtin-remark s                        <
//
//  ============================================================

EOF
}

# ---------------------------------
#  process command-line
# ---------------------------------

mode=""

case "$1" in
    --help|--hel|--he|--h|-help|-hel|-he|-h|"-?")
        display_help
        exit $E_SUCCESS
        ;;
    --show|-s)
        print_example
        exit $E_SUCCESS
        ;;
    --include|-i)
        mode="include"
        ;;
    "")
        mode="normal"
        ;;
    *)
        echo "$SCRIPT: try --help for usage"
        exit $E_USAGE
        ;;
esac

# ---------------------------------
#  confirm_base()
# ---------------------------------

function confirm_base
{
    test -f "$MAINCC" && return 0
    echo "$SCRIPT: context failure: file not found (as proxy for base directory): $MAINCC "
    return 1
}

# ---------------------------------
#  identify_headers
# ---------------------------------

headers=""                              # unit headers (as opposed to stand-alone headers)

function identify_headers
{
    local header
    local file

    for file in $( make ccs 2>/dev/null )    # target "ccs" is supported by 'makefile'
      do
      header="${file%.cc}.h"            # swap ".cc" for ".h"
```

```
      test -f "$header" || continue     # abandon loop if file not found
      headers="$headers $header"        # concatenate
    done
    test -n "$headers" && headers="${headers:1}"  # strip leading space

    test -n "$headers" && return 0      # headers found
    echo "$SCRIPT: no headers found"
    return 1                            # no headers found
}

# ---------------------------------
#  active code
# ---------------------------------

#  Notes for both modes
#
#      entry header must begin, as a absolute minimum "//====XEDOC===="
#      exit header must begin, ditto                  "//===="
#
#      more precisely:
#
#        RS            record separator (default "\n")
#        ORS           output record separator (default "\n")
#        $1            first field
#        $0            entire record
#        FILENAME      current input file
#        [[:space:]]   ASCII space or horizontal tab chars
#
#      the 'sed' call strips leading C++ comments from "//" to "//  "
#      the 'grep' call omits lines containing the pattern shown from identified records
#
#      CAUTION: note the different escaping requirements for 'regex' postfix
#      operator '?' in 'gawk' and 'sed'
#
#      CAUTION: RS as a 'regex' only in 'gawk' and 'mawk' but not 'awk'
#
#  Additional notes for the 'include' mode
#
#      the second 'gawk' call converts real newlines to "\n" strings
#      the final 'sed' call converts """ strings to "\"" strings, the 'g' flag is essential

# confirm run from base directory

confirm_base || exit $E_NO_MAIN

# generate file list

identify_headers || exit $E_NO_HEADERS_IDENTIFIED

# main processing

case "$mode" in

    normal)

        loops=0                         # number of headers processed
        for file in $headers            # CAUTION: no soft or strong quotes here
          do
          let "++loops"
          gawk 'BEGIN { RS = "//[[:space:]]*====="; ORS = "" } \
              $1 ~ /XEDOC/ { print $0; printf("    header: %s\n\n", FILENAME) } \
              END { }' \
              "$file" | \
              sed 's/\/\/[[:space:]]\?[[:space:]]\?//' | \
              grep --invert-match "[[:space:]]*XEDOC[[:space:]]*=====*"
        done

        {
            echo
            echo "---"
            echo "$SCRIPT: headers proceed   : $loops"
            echo "$SCRIPT: working directory : $(pwd -P)"
            echo
        } >&2                           # code block redirected to stderr (to facilitate dumping)
        ;;

    include)

        echo -n "\""
        for file in $headers            # CAUTION: no soft or strong quotes here
          do
```

```
          gawk 'BEGIN { RS = "//[[:space:]]*====="; ORS = "" } \
              $1 ~ /XEDOC/ { print $0; printf("    header: %s\n\n", FILENAME) } \
              END { }' \
              "$file" | \
              sed 's/\/\/[[:space:]]\?[[:space:]]\?//' | \
              grep --invert-match "[[:space:]]*XEDOC[[:space:]]*=====*" | \
              gawk 'BEGIN { RS = "\n"; ORS = "\\n" } { print $0 }' | \
              sed 's/"/\\"/g'
        done
        echo "\""
        ;;

esac

# ---------------------------------
#  housekeeping
# ---------------------------------

exit $E_SUCCESS

# ---------------------------------
#  junk
# ---------------------------------

# BEGIN { getline; } can be useful

#  $Id: xedoc 3496 2009-10-09 08:39:32Z robbie $
#  end of file
```

```
#! /bin/bash

#  file-purpose     : interface for xeona makefile with local redefines
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Thu 12-Apr-2007 21:25 UTC
#  file-status      : working
#  file-keywords    : xeona

#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/scripts/mach $

#  TODO: update xeona exit codes as these become known
#        perhaps: valgrind --tool=massif ./binary
#        will mach take multiple macros from option -D ?
#
#  work on 'set' versus 'shopt' versus PIPESTATUS
#  do 0-32 return code tricks

#  ------------------------------------------------------------
#  Motivation
#
#      The principal motivation behind this script is
#      the desire to avoid:
#
#        * a large number of unit-test makefiles
#        * the continual hand editing of makefiles
#
#      Now just this script and the project makefile
#      are all that are necessary.  And only the source
#      file list in the project makefile needs updating
#      as new *.cc files are added to the application
#      build.  Header dependencies are handled
#      automatically by the 'fastdep' utility.  While
#      preprocessor and compiler flags can be pumped in
#      using the various command-line options provided
#      in this script.
#
#  Roles
#
#      This file can be used either to:
#
#        * build the primary application
#        * unit-test an individual source file
#
#      The choice is made based on the filename
#      supplied as the command-line argument.  See
#      '--help' for details.
#
#      It is assumed that 'common.h' and 'logger.h' are
#      always hash-included in the source file under
#      unit-testing.
#
#  Script test option
#
#      Note the '-t' option prevents exiting on error.
#      This, of course, is likely to lead to an
#      avalanche of further errors.
#
#  Additional scripts
#
#      This script is the lowest script in the
#      following script call chain:
#
#          mach < pruf < jede < ganz
#
#      See "ganz --help" for more details.
#
#      After reverting to a single codebase in mid-2008,
#      'jede' and 'ganz' are now obsolete.
#
#  References
#
#      Robbins, Arnold.  2005.  GDB pocket reference.
#        O'Reilly Media, Sebastopol, CA, USA.  ISBN
#        0-596-10027-2.
#
#  ------------------------------------------------------------

# -------------------------
#  preamble
# -------------------------
```

```
TESTSDIR="xeona-mach"                   # subdirectory for testing binaries

host=$(hostname --short)
case "$host" in
    hinau) SVN_BASE="$HOME/synk/xeona/svn2/futz/trunk"  ;;  # svn version base for the purposes of ⬋
this script
    sojus) SVN_BASE="$HOME/synk/xeona/svn/futz"         ;;  # svn version base for the purposes of ⬋
this script
    *)     SVN_BASE="$HOME/synk/xeona/svn/futz"         ;;  # svn version base for the purposes of ⬋
this script
esac

REPOS="$HOME/svn-root/xeona/futz"       # repository path (for comment in final reporting only)
PRUF="pruf"                             # additional script (for comment in '--help' message only)

UNIT_TEST_LOCAL="machunits"             # local branch-specific unit-test lists (separate file for ⬋
ease of maintenance)
UNIT_TEST_LISTS="mach.units.sh"         # universal unit-test lists (used if local file not present)

DEFAULT_BINARY="xeona.mach"             # default application name if none given
APP_ARGS=""                             # arguments to be passed to the binary
APP_ARGS="--report 5"                   # increase reporting
APP_ARGS="--report 5 --inbuilt 6 --exittrip 2"    # plus generate and use default 'inbuilt.xem' at ⬋
'exittrip' 2

SPECIAL_MAKEFILE=""                     # if a nonstandard makefile name is to be used
UTEST="ut"                              # tester tag in form: unitname.${UTEST}[0-9].cc
COMMON="common"                         # stem for common header: $COMMON.cc
MAINCC="main.cc"                        # name of main file

CXXDEBUG="-g"                           # the most basic and portable compiler debug option
CXXDEBUG="-g3 -gdwarf-2"                # suggested by Robbins (2005 p4) when debugging with GDB
CXXDEBUG="-ggdb3"

CXXLDPROF="-p"                          # the most basic and portable compiler profile info option
CXXLDPROF="-pg"                         # for the GNU profiler 'gprof'

VALGRIND="valgrind"                     # memory check utility
VALGRIND_OPTS1="--leak-check=full --show-reachable=yes" # extra options, '--verbose' is very loud

XEDOC="xedoc"                           # script generating xedoc string
XEDOC_OUTPUT="a/xedocs.txt"             # as per 'common.cc'
XEDOC_ASUSED="a/xedocs.as-used.txt"     # primarily for bug hunting

XMOK="xmok"
XMOK_ARGS="--"

CTRLC=130                               # system-dependent return for ctrl-c interrupt
ABORT=134                               # system-dependent return for terminate (uncaught throw)
SEGFAULT=139                            # system-dependent return for segmentation fault

SCRIPT=$( basename "$0" )               # this file
COUNTFILE="${SCRIPT}countfile"          # used to count calls for display in console
SVNVERFILE="lastbuild"                  # used to record svn details for last build

TEMPSTUB=$( mktemp -u _XXXXXXXX )       # filename stub for safe temporary filenames

# ---------------------------------
#  default overrides as needed
# ---------------------------------

#SPECIAL_MAKEFILE="makefile_1.33.1"     # if a nonstandard makefile name is to be used

# ---------------------------------
#  exit statii
# ---------------------------------

# normal exit is 0-31, trivial exit is 100+, abnormal exit is 200+

X_SUCCESS=0
X_COMPILE_FAIL=1
X_COMPILE_WARN=2
X_RUN_FAIL=4
X_MEMORY_ERROR=8
X_MEMORY_LEAK=16
X_XMOK_FAIL=32
X_XEDOCS_FAIL=64

E_SUCCESS=100
E_USAGE=102
```

```
E_MISSING_FILES=201
E_RELEASE_ISSUES=202
E_UPDATE_ISSUES=203
E_MAKE_FAIL=204
E_CODING_ERROR=205
E_REFUSING_COMMON=206
E_RM_FAIL=207
E_MISSING_SUBDIR=208

# ---------------------------------
#  initialization
# ---------------------------------

xstat=$X_SUCCESS    # overall status

gccfail=0           # else compile failure (not used)
gccwarn=0           # else compile warnings
runret=0            # else execute failure
valerror=0          # else memory errors
valleak=0           # else memory leakage

nobin=0             # else no binary
emptybin=0          # else zero-byte binary

# ---------------------------------
#  display_help()
# ---------------------------------

function display_help
{
local release="xeona.r000"              # set default
local custom="xeona.custom"             # set default
local unit="unit"                       # set default

local core=""                           # default
local host=$(hostname --short)
case "$host" in
    hinau) core="4";                    # four core processor
esac

case "$1" in
    "")      :            ;;            # blank, do nothing
    xeona.)  :            ;;            # unacceptable, do nothing
    xeona.*) custom="$1"  ;;            # overwrite
    *.*)     :            ;;            # unacceptable, do nothing
    *)       unit="$1"    ;;            # overwrite
esac

local xtra="$2"                         # optional second argument
local extra=""
test -n "$xtra" && extra=" (note -H removes -${xtra} from some examples)"

local fs3=""                            # Boost.Filesystem 1.43 and earlier (F could also be used)
local fs3=""                            # Boost.Filesystem 1.45 and beyond  (F could also be used)
local fs3="F3"                          # Boost.Filesystem 1.44 (the F is optional, the 3 is not)

local opts="[-D MACRO] [-a -b -d -f -K|k -m -n -P -p -q -r -S|s -T -t -u -V|v -x]"
local lin1="a prior clean svn"
local lin2="valgrind memory checker"
local lin3="build and check using built-in defaults"
local lin4="keep temporary files, add debug reporting, sometimes refuse to die"
local lin5="exclude unit test friendships and similar"
local lin6="best with -fx, compiler may still complain"
local lin7="use -f (fresh compile) if the preprocessor or compiler flags are changed or different ⬋
files are called"

echo
echo "       usage: $SCRIPT  $opts  [binary-name]  (assumes xeona*)"
echo "              $SCRIPT  $opts   unit-name     (assumes *.${UTEST}[0-9].cc)"
echo "              $SCRIPT   --                          $lin3 (including $DEFAULT_BINARY)"
echo "              $SCRIPT   -c          [subdir-name]  clean only (in subdir if given)"
echo "              $SCRIPT   -h  --help  [file-name]    this message (duly modified)"
echo "     options:  -a     alert               enable compile and final beeps (but -b takes ⬋
precedence)"
echo "               -b     no beep             kill beep including double beep on failed make and run ⬋
calls"
echo "               -c     clean only          prompt to clean up relevant 'subversion safe' files ⬋
and quit"
echo "               -D     MACRO               apply _MACRO preprocessor macro (can only use once)"
echo "               -d     not debug           omit _XDEBUG preprocessor macro (mostly to report ⬋
```

```
less)"
echo "               -e     ensure xedocs       write out and run xedocs suite"
echo "               -f     fresh compile       delete all object files and build afresh"
echo "               -F     Boost.Filesystem    define BOOST_FILESYSTEM_NO_DEPRECATED for ⬀
<boost/filesystem>"
echo "               -g     no GNU              apply g++ -std=c++98 to exclude GNU extensions"
echo "               -h     help                display usage message and quit (takes ⬀
precedence)$extra"
echo "               -K     klean all           clean up all created files including binary (will also ⬀
build)"
echo "               -k     klean some          clean up all created files except binary (will also ⬀
build)"
echo "               -m     meyers              apply g++ -Weffc++ for Scott Meyers' style guides ⬀
(note -z|Z)"
echo "               -n     dry run             print compiler calls and quit ($lin6)"
echo "               -o     okay                call $XMOK but only if binary built anew this time"
echo "               -P     profiling info      apply $CXXLDPROF to support profiling (non-release ⬀
only) (leaks memory too)"
echo "               -p     pedantic            apply g++ -pedantic for more fussy warnings (note -Z)"
echo "               -q     no -Wall            suppress normal warnings"
echo "               -r     release build       optimize code and use special name (also requires -df ⬀
and $lin1)"
echo "               -S     logger only         redirect stdout stream to /dev/null (except -V|v)"
echo "               -s     screen only         redirect logger and stderr streams to /dev/null ⬀
(except -V|v)"
echo "               -T     utest less          omit _XUTEST preprocessor macro ($lin5)"
echo "               -t     test                $lin4"
echo "               -u     undeclared okay     apply g++ -fpermissive to warn, rather than error, on ⬀
undeclared names"
echo "               -V     valgrind less       use $lin2 (but -x takes precedence)"
echo "               -v     valgrind more       use $lin2 with extra options (but -x takes precedence)"
echo "               -x     no execute          do not run binary"
echo "               -Z     no filter           omit all warning screens (under -m and -p)"
echo "               -z     some filter         omit \"instantiated from\" screen (under -m)"
echo "               -3     Boost.Filesystem 3  define BOOST_FILESYSTEM_VERSION 3 for ⬀
<boost/filesystem>"
echo "               -4     multicore           apply make option --jobs=4 (faster but can interleave ⬀
outout)"
echo "     purpose: interface xeona project makefile with local redefines"
echo "     caution: $lin7"
echo "  versioning: embedding   : subversion release number embedded in binary if the given svn is ⬀
clean"
echo "              svn basis   : $SVN_BASE"
echo "              current dir : $( pwd -P )"
echo "              version     : run \$ svnversion $SVN_BASE"
#test -n $( which svnversion ) && echo "              version     : $( svnversion $SVN_BASE )"
echo "    settings: default binary name          : $DEFAULT_BINARY"
echo "              default application args      : $APP_ARGS"
echo "              release binary name pattern   : xeona.r000"
echo "              test subdirectory             : $TESTSDIR/"
echo "              count file name (can delete)  : $COUNTFILE"
echo "              unit-test tag                 : $UTEST"
echo "              common header                 : $COMMON.h"
echo "              valgrind extra options (-v)   : $VALGRIND_OPTS1"
echo "              local unit-test lists         : $UNIT_TEST_LOCAL"
echo "    examples: clean up current dir safely (uses svn info)       : \$ $SCRIPT -c"
echo "              clean up current branch safely (based on above)   : \$ $PRUF -c"
echo "              basic build and run                               : \$ $SCRIPT --"
echo "              basic build but run with --opt                    : \$ $SCRIPT -x${fs3} $custom && ⬀
$custom --opt"
echo "              release build (requires a prior clean svn)        : \$ $SCRIPT -df${fs3}rx"
echo "              fresh with full warn, memtest, xedoc, okay, alert : \$ $SCRIPT ⬀
-${xtra}${fs3}${core}mpveoa"
echo "              style police with no output filtering (noisy!)    : \$ $SCRIPT ⬀
-${xtra}${fs3}${core}mpZ"
echo "              unit-test with audible notification               : \$ $SCRIPT ⬀
-${xtra}${fs3}${core}a    $unit"
echo "              unit-test with everything (although not -t)       : \$ $SCRIPT ⬀
-${xtra}${fs3}${core}mpva $unit"
echo

return 0
}

# ---------------------------------
#  screen command-line
# ---------------------------------

cline="$*"                              # grab original command-line for reporting

# the strings "--", "-", and "" (no options or arguments) are also caught as required
```

```
case "$1" in
    --help|--hel|--he|--h|-help|-hel|-he|-h|"-?")
        display_help "$2" "f"
        exit $E_SUCCESS
        ;;
    --H|-H)
        display_help "$2"
        exit $E_SUCCESS
        ;;
    help|-|"")
        echo "$SCRIPT: incorrect usage, try --help"
        exit $E_USAGE
        ;;
    --)                                 # fall thru
        :
        ;;
esac

# ------------------------------
#  process options
# ------------------------------

# set defaults

alert_flag=0          # -a
nobeep_flag=0         # -b
clean_only_flag=0     # -c
CPPMACRO=""           # -DMACRO
debug_omit_flag=0     # -d (same effect as 'removing' -D_XDEBUG)
xedocs_flag=0         # -e
fresh_flag=0          # -f
filesys_flag=0        # -F
no_gnu_flag=0         # -g
klean_flag=0          # -k
Klean_flag=0          # -K
meyers_flag=0         # -m
dryrun_flag=0         # -n
xmok_flag=0           # -o
profile_flag=0        # -P
pedantic_flag=0       # -p
quiet_flag=0          # -q
release_flag=0        # -r
logger_only_flag=0    # -S
screen_only_flag=0    # -s
test_flag=0           # -t
utest_omit_flag=0     # -T
valgrind_extd_flag=0  # -V
valgrind_flag=0       # -v
permissive_flag=0     # -u
no_execute_flag=0     # -x
nofilter_flag=0       # -Z
lessfilter_flag=0     # -z
filesys_v3_flag=0     # -3
fourcore_flag=0       # -4

# process options

while getopts ":abcD:defFgKkmnoPpqrSsTtuVvxZz34" option  # CAUTION: the leading : should be correct
do
    case "$option" in
        a)  alert_flag=1                ;;
        b)  nobeep_flag=1               ;;
        c)  clean_only_flag=1           ;;

    #  note that this next option must have an associated argument,
    #  otherwise it falls through to the default.

        D)  CPPMACRO="$OPTARG"          ;;
        d)  debug_omit_flag=1           ;;
        e)  xedocs_flag=1               ;;
        f)  fresh_flag=1                ;;
        F)  filesys_flag=1              ;;
        g)  no_gnu_flag=1               ;;
        K)  Klean_flag=1; klean_flag=1  ;;
        k)  klean_flag=1                ;;
        m)  meyers_flag=1               ;;
        n)  dryrun_flag=1               ;;
        o)  xmok_flag=1                 ;;
        P)  profile_flag=1              ;;
        p)  pedantic_flag=1             ;;
```

```
        q)  quiet_flag=1                ;;
        r)  release_flag=1              ;;
        S)  logger_only_flag=1          ;;
        s)  screen_only_flag=1          ;;
        T)  utest_omit_flag=1           ;;
        t)  test_flag=1                 ;;
        u)  permissive_flag=1           ;;
        V)  valgrind_flag=1             ;;
        v)  valgrind_extd_flag=1        ;;
        x)  no_execute_flag=1           ;;
        Z)  nofilter_flag=1             ;;
        z)  lessfilter_flag=1           ;;
        3)  filesys_v3_flag=1           ;;
        4)  fourcore_flag=1             ;;
        *)
            echo "$SCRIPT: incorrect usage, try --help"
            exit $E_USAGE
            ;;
    esac
done
shift $(($OPTIND - 1))

#  the above decrements the argument pointer so it points to next
#  argument, hence $1 now references the first non-option supplied
#  on the command-line, in the event that substantive arguments
#  were given

SUBSTANTIVE_ARG="$1"

# ---------------------------------
#  alert()
# ---------------------------------

function alert
{
    local count=$1

    test $nobeep_flag -eq 1 && return 1 # -b (no beep) always takes precedence

    test $alert_flag -eq 1 && sleep 1   # to provide a gap between consecutive calls
    if [ $( which beep ) ]              # CAUTION: for test, no -n and no soft-quotes
        then
        if [ $count -gt 0 ]             # if positive .. beep normal
            then
            beep -r $count
        elif [ $count -lt 0 ]           # if negative .. beep low
            then
            let "count *= -1"           # positive argument required
            beep -r $count -f 261.6
        else
            :                           # count is zero, do nothing
        fi
    else
        echo -ne "\a"                   # fall back beep
    fi
    return 0
}

# -------------------------------
#  report()
# -------------------------------

function report
{
    local tab=-18                       # the "-" means left justify
    case "$#" in                        # argument count
        0) printf "\n"                                     ;;
        1) printf "$SCRIPT: %s\n" "$1"                    ;;
        2)
           # coloration
           if [[ "$1" == "PROBLEM" && "$TERM" != "dumb" ]]
               then
               local tag=$( echo -e "\033[01;35mproblem\033[00m" )
               printf "$SCRIPT: $tag            : %s\n" "$2"
           else
               printf "$SCRIPT: %${tab}s : %s\n" "$1" "$2"
           fi
           ;;
        *)
            report "CODING ERROR" "more than two arguments given for report()"
            exit $E_CODING_ERROR
```

```
            ;;
    esac
    return 0
}

# ---------------------------------
#  deport()
# ---------------------------------

# debug reporting as interface to 'report'

function deport
{
    test $test_flag -eq 0 && return 1   # skip if -t (test) option not set

    case "$#" in                        # argument count
        0) report            ;;
        1) report "$1"       ;;
        2) report "$1" "$2"  ;;
    esac
}

# -------------------------
#  check_for_utility()
# -------------------------

function check_for_utility
{
    local utility="$1"
    if [ -z $( which "$utility" ) ]     # soft-quotes around shell call not needed
    then
        report "ERROR" "$utility utility not found"
        missing_flag=1
        return 1
    fi
    return 0
}

# ---------------------------------
#  foreign_machine()
# ---------------------------------

function foreign_machine
{
    local BOOST_CPATH=""                          # hardcoded here
    BOOST_CPATH="/usr/local/include/boost-1_39"   # hardcoded here
    BOOST_CPATH="/usr/local/include/boost-1_38"   # hardcoded here

    local ORIGINAL_NODE="sojus"                        # original development machine

    local node=$( uname --nodename )
    test "$node" == "$ORIGINAL_NODE" && return 0

    report "foreign node" "$node"                      # foreign machine name

    test -f "$MAINCC" ||                               # main file (probably main.cc) exits and is ⬈
local
    {
        report "PROBLEM"  "abandoning foreign_machine settings as not in main file ($MAINCC) directory"
        report
        return 1
    }

    report "CAUTION" "the following settings and exports are local to this script"

    SVN_BASE=$( pwd -P )
    report "svn base thus" "$SVN_BASE"

    local cpath="$BOOST_CPATH"
    test -n "$CPATH" && CPATH="$CPATH:"
    export CPATH="$CPATH$cpath"
    report "CPATH now contains" "$cpath"

    local path="$( pwd -P )/../scripts"                # arguably better
    local path="$( dirname $( pwd -P ) )/scripts"      # a bit rough but works
    export PATH="$PATH:$path"
    report "PATH now contains" "$path"
    deport "PATH comprises" "$PATH"

    report
    return 0
```

```
}

# ------------------------------
#  final_line()
# ------------------------------

# standardized final line with coloration

function final_line
{
    # prepare exit status

    local retstr=""
    retstr="-- okay ---"                  # alternative reportage
    retstr="-----------"
    local fill="+"
    local C=$fill; test ${xreduce[0]} -ne 0 && C="C"  # Compile
    local W=$fill; test ${xreduce[1]} -ne 0 && W="W"  # Warn
    local X=$fill; test ${xreduce[2]} -ne 0 && X="X"  # eXecute
    local E=$fill; test ${xreduce[3]} -ne 0 && E="E"  # memErrors
    local L=$fill; test ${xreduce[4]} -ne 0 && L="L"  # Leakage
    test $xstat -ne 0 && retstr=" $C$W$X$E$L "

    # prepare binary name

    local binstr=" $BINARY "
    test $clean_only_flag -ne 0 && binstr="$cleanstr" # for -c (clean only) option
    while [ ${#binstr} -lt 30 ]; do binstr="-$binstr"; done # pad with leading '-'

    # prepare directory name

    local locdir=$( basename $( pwd -P ) )
    test -n "$dir" && locdir="$dir"     # overwrite when cleaning (can comment out)
    dirstr=" $locdir "
    while [ ${#dirstr} -lt 30 ]; do dirstr="$dirstr-"; done # pad with trailing '-'

    # counter string
    local cntstr=" $countstr "

    # create final line
    # CAUTION: printf requires the '--' option because the format string starts with a '-'

    local line=""
    printf -v line -- "---%s----------------------------%s----------%s----------%s------" \
                      "$dirstr" "$binstr" "$retstr" "$cntstr"

    # colorize line
    # attributes 00=normal 01=bold / color 30=black 31=red 35=magenta 36=cyan

    test "$TERM" != "dumb" && line=$( echo -e "\033[00;36m${line}\033[00m" )

    # report

    report
    report "$line"
    report
}

# ---------------------------------
#  call_xedoc()
# ---------------------------------

# view output with: $ echo -e "$( cat a/xedocs.as-used.txt )" | less

function call_xedoc
{
    local mode="$1"

    case "$mode" in
        refresh)
            test -f "$XEDOC_OUTPUT" && rm -f "$XEDOC_OUTPUT"   # remove existing file, probably just ""
            test -f "$XEDOC_ASUSED" && rm -f "$XEDOC_ASUSED"   # remove existing file, probably full
            $XEDOC --include > "$XEDOC_OUTPUT"                 # create new file in correct format
            cp "$XEDOC_OUTPUT" "$XEDOC_ASUSED"                 # primarily for bug hunting
            report "$XEDOC_OUTPUT" "xedocs file refreshed"     # report
            return 0
            ;;
        nullify)
            test -f "$XEDOC_OUTPUT" && rm -f "$XEDOC_OUTPUT"   # remove existing file
            echo "\"\"" > "$XEDOC_OUTPUT"                      # create new file in null format
            report "$XEDOC_OUTPUT" "xedocs file nullified"     # report
```

```
            return 0
            ;;
        *)
            report "CODING ERROR" "unsupported argument given for call_xedoc(): $mode"
            return 1
            ;;
    esac
}

# ---------------------------------
#  delete_binary()
# ---------------------------------

function delete_binary
{
    rm --force "$BINARY"
    report "binary deleted" "$BINARY"
}

# ---------------------------------
#  reduce()
# ---------------------------------

# for instance '27' reduces to '1 2 0 8 16'

declare -a xreduce                      # -a is array
declare xsum

function reduce
{
    num="$1"                            # integer being operated on

    xsum=                               # reset summation
    xreduce=()                          # reset results array

    local base                          # power of two divisor
    let "base = 2 ** $reduce_exp"

    let "max = 2 * $base"

    test $num -ge $max && return 1      # too large an integer
    test $num -lt 0    && return 1      # a negative-definite integer

    local index=$reduce_exp
    while [ $base -gt 0 ]
      do
      if [ $num -ge $base ]
          then
          let "num = $num % $base"      # note % remainder operator
          xreduce[$index]=$base
      else
          xreduce[$index]=0
      fi
      let "base /= 2"
      let "xsum += ${xreduce[$index]}"
      let "index -= 1"
    done

    # TOFIX: remove in due course, committed r3025, 06-Jul-2009
    # never gets here when the xedocs code spits the dummy
    # echo
    # echo "$SCRIPT: temporary adhoc reporting: xsum = $xsum, xreduce = ${#xreduce[@]} elements"

    return 0
}

# ---------------------------------
#  controlled_exit()
# ---------------------------------

function controlled_exit
{
    case "$release_flag" in
        1) helpful_dump_calls ;;
    esac

    exit_code=$1

    reduce_exp=5                        # reduce base exponent, 5 means 0 thru 63 valid
    reduce $xstat                       # function call
    test $xstat -eq $xsum || report "CODING ERROR" "xstat and xsum differ: $xstat $xsum"
```

```
    let "warn = $xstat + nobin + emptybin"

    deport
    deport "xstat"   "$xstat"
    deport "xreduce" "${xreduce[*]}"    # soft-quoting okay in this case

    test $warn         -ne 0 && report
    test ${xreduce[0]} -ne 0 && report "PROBLEM" "compile failure    [${xreduce[0]}]"
    test ${xreduce[1]} -ne 0 && report "PROBLEM" "compile warnings   [${xreduce[1]}]"
    test $nobin        -ne 0 && report "PROBLEM" "no binary file     [-]"
    test $emptybin     -ne 0 && report "PROBLEM" "empty binary file  [-]"
    test ${xreduce[2]} -ne 0 && report "PROBLEM" "execute failure    [${xreduce[2]}]"
    test ${xreduce[3]} -ne 0 && report "PROBLEM" "memory errors      [${xreduce[3]}]"
    test ${xreduce[4]} -ne 0 && report "PROBLEM" "memory leakage    [${xreduce[4]}]" # alignment is ⬋
okay
    test ${xreduce[5]} -ne 0 && report "PROBLEM" "test suite issues [${xreduce[5]}]" # alignment is ⬋
okay
    test $warn         -ne 0 && alert 2

    case "$exit_code" in
        $E_SUCCESS)               meaning="script success"                ;;
        $E_USAGE)                 meaning="script usage issue"            ;;
        $E_MISSING_FILES)         meaning="files missing"                 ;;
        $E_RELEASE_ISSUE)         meaning="release build issues"          ;;
        $E_UPDATE_ISSUE)          meaning="update issues"                 ;;
        $E_MAKE_FAIL)             meaning="make returned fail"            ;;
        $E_CODING_ERROR)          meaning="coding error -- check $SCRIPT" ;;
        $E_REFUSING_COMMON)       meaning="unwilling to unit test common" ;;
        $E_MISSING_SUBDIR)        meaning="subdirectory not found"        ;;
        0)                        meaning="clean run"                     ;;
        1|2|4|8|16|32|64)         meaning="single problem"                ;;
        [1-9]|[1-9][0-9]|1[0-1][0-9]|12[0-7])     # 1-127, note prior capture
                                  meaning="multiple problems"             ;;
        *)                        meaning="undocumented script return"    ;;
    esac

    report
    report "old/new svns" "$svnver / $( svnversion $SVN_BASE )"
    report "command-line" "\$ $SCRIPT $cline"
    report "binary" "$BINARY"
    local seconds=$SECONDS
    test $seconds -ne 1 && plural="s"
    report "elapsed time" "$seconds second$plural"
    local elapsed=$SECONDS
    test $( which hms ) && elapsed=$( hms $elapsed )   # 'hms' is a my (user-local) utility
    report "elapsed time" "$elapsed"
    report "complete"     "script exit $exit_code ($meaning)"
    [[ $alertflag -eq 1 && $warn -eq 0 ]] && alert 1 # beep if -a (alert) and no warnings
    final_line $exit_code               # print final line
    exit $exit_code
}

# ------------------------------
#  countfile()
# ------------------------------

# update run counting with COUNTFILE

function countfile
{
    test -f "$COUNTFILE" || echo "0" > "$COUNTFILE"
    count=$( cat "$COUNTFILE" )
    test $count -ge 9999 && count=0     # rollover as required
    let "count++"
    echo "$count" >| "$COUNTFILE"       # force clobber
    printf -v countstr "%04d" $count    # create four-char zero-padded string
}

# ---------------------------------
#  svn_ver_and_state()
# ---------------------------------

svnver=""
svn_flag=""

function svn_ver_and_state
{
    svnver=$( svnversion $SVN_BASE 2>/dev/null )

    # if svnver contains : then an update is needed
```

```
    # if svnver contains M then a commit and update is needed

    svn_flag="clean"
    test $( expr index "$svnver" ":" ) -ne 0 && mess="update"
    test $( expr index "$svnver" "M" ) -ne 0 && mess="commit and update"
    test -n "$mess" && svn_flag="stale"
}

# ---------------------------------
#  check_maincc()
# ---------------------------------

function check_maincc
{
    test -f "$MAINCC" && return 0       # main file (probably main.cc) exits and is local
    report "PROBLEM" "an app build requires a main file but none found: $MAINCC"
    report "check" "is this directory correct or should you be testing units instead?"
    test $test_flag -eq 0 && controlled_exit $E_MISSING_FILES       # exit gracefully
    report "note" "carrying on irrespective under option -t (test)"
    report
    return 1
}

# ---------------------------------
#  helpful_dump_calls()
# ---------------------------------

# $ svnadmin dump ~/svn-root/xeona/futz/ -r 2499 > xeona-2499.svndump
# $ tar -cvf xeona-2499.svndump.tar xeona-2499.svndump
# CAUTION: 'svnadmin' utility does not take URLs

function helpful_dump_calls
{
    local dumpfile="xeona-$svnver.svndump"
    local cpopts="--preserve=mode,ownership,timestamps,link"
    local rbindir="xeona-rbin"
    if [ -d "$REPOS" ]
        then
        report
        report "dump call 1" "\$ svnadmin dump $REPOS -r $svnver > $dumpfile"
        report "dump call 2" "\$ tar -cvf $dumpfile.tar $dumpfile"
        if [ -d "$rbindir" ]
            then
            report "copy call 3" "\$ cp $cpopts $BINARY $rbindir"
        fi
    else
        report
        report "dump call PROBLEM"   "directory not found: $REPOS"
    fi
    return 0
}

# -------------------------
#  confirm utilities
# -------------------------

report
report "script commencing"
report

missing_flag=0                          # reset in check_for_utility() if utility not found
check_for_utility "fastdep"             # used in the makefile
check_for_utility "svnversion"          # used in this file
check_for_utility "valgrind"            # used in this file
check_for_utility "gawk"                # used in this file
check_for_utility "$XEDOC"              # used to create 'xedocs' file
check_for_utility "$XMOK"               # used to run various binary tests
test $missing_flag -ne 0 && report      # nothing further happens

# ---------------------------------
#  active code
# ---------------------------------

foreign_machine                         # if non-original development machine
countfile                               # increment countfile
svn_ver_and_state                       # set 'svnver' and 'svn_flag'

# ---------------------------------
#  test status reporting
# ---------------------------------
```

```
deport "TEST" "option -t (test) set"
if [ $clean_only_flag -eq 1 ]           # disable -t (test) under -c (clean only) option
    then
    deport "note" "nothing useful to be reported under -c (clean only)"
    deport
    test_flag=0                         # reset test flag
fi
deport "note" "temporary files kept, debug reporting added"
deport

# ---------------------------------
#  screen MACRO definition
# ---------------------------------

# MACRO means macro arguments to be passed to the compiler

note1="check command-line"              # just used here

if [ $( expr index "$CPPMACRO" "-" ) -ne 0 ]      # one-based indexing, has a - somewhere
    then
    report "ERROR" "macro '-D $CPPMACRO' cannot be successfully passed to g++ ($note1)"
    controlled_exit $E_USAGE
fi

if [ "${CPPMACRO:0:5}" == "xeona" ]     # cannot use xeona* as a preprocessor macro
    then
    report "ERROR" "will not accept macro '-D $CPPMACRO' starting 'xeona' ($note1)"
    controlled_exit $E_USAGE
fi

#  the following code is commented out because the
#  usage rule requiring an explicit binary name is now
#  deemed unnecessary
#
#  if [[ -n "$CPPMACRO" && -z "$1" ]]      # must supply an explicit filename
#      then
#      note2="without an explicit binary name"
#      report "ERROR" "will not accept macro '-D $CPPMACRO' $note2 ($note1)"
#      controlled_exit $E_USAGE
#  fi

# -------------------------
#  process filenames
# -------------------------

STEM="$SUBSTANTIVE_ARG"                 # substantive command-line argument, may be empty

test -z "$STEM" && STEM="$DEFAULT_BINARY"    # use default name (xeona.mach)

STEM=${STEM%".cc"}                      # strip trailing ".cc" if supplied
STEM=${STEM%.${UTEST}[0-9]}             # strip trailing unit test extension ".ut[0-9]" if present

case "$release_flag" in                 # special naming convention for release builds
    1)
        svnver=$( svnversion $SVN_BASE 2>/dev/null )   # should be an integer string
        BINARY="xeona.r$svnver"
        ;;
    *)
        BINARY=$STEM
        ;;
esac

case "$BINARY" in
    xeona*)
        type="app"                      # use full sources list as per makefile
        test $clean_only_flag -eq 0 && check_maincc  # check for main file
        ;;
    *)
        type="unit"                     # use unit-test conventions
        ;;
esac

# ---------------------------------
#  clean only option
# ---------------------------------

case "$clean_only_flag" in
    1)
        svnver=$( svnversion $SVN_BASE 2>/dev/null )   # for end-of-script reporting

        dir=$( basename $( pwd -P ) )   # for reporting purposes only
```

```
        if [ -n "$1" ]                  # subdir argument given (may be "." "./" "sdir" "sdir/")
            then
            SDIR=$( basename "$1" )     # subdirectory to be cleaned, remove any trailing slash
            if [ "$SDIR" != "." ]       # no need to further process a "."
                then
                test -d "$SDIR" || controlled_exit $E_MISSING_SUBDIR
                dir=$dir/$SDIR          # add the root directory for reporting purposes
                cd "$SDIR"              # change to selected directory
            fi
        fi

        report "$dir" \
          "cleaning up '*.o' (object), 'svn ?' (not ignored), and some _*.txt ($SCRIPT temp) files"

        # generate file list in three parts

        list1=$( ls -1 *.o 2>/dev/null )
        list2=$( svn status --verbose --no-ignore --non-recursive \
            | grep '?' \
            | sed 's/^? *//' \
            | sort )
        list3=$( ls -1 _????????-???.txt 2>/dev/null )

        # concatenate using printf

        list=""
        test -n "$list1" && printf -v list "${list}%s\n\n" "$list1"
        test -n "$list2" && printf -v list "${list}%s\n\n" "$list2"
        test -n "$list3" && printf -v list "${list}%s\n\n" "$list3"

        # report and exit if required

        echo
        if [ -n "$list" ]
            then
            echo -n "$list"             # -n means one less final newline
        else
            echo "(no matches)"
            controlled_exit $E_SUCCESS
        fi

        # seek confirmation

        echo -n "  $SCRIPT: enter 'y' to delete: "
        read response
        if [ "$response" != "y" ]
            then
            report
            report "abandoning clean up"
            cleanstr=""                 # used in final_line reporting
            controlled_exit $E_SUCCESS
        fi

        # delete

        list=$( echo $list )

        cleanexit=$E_SUCCESS
        rmcount=0                       # file removal counter
        for file in $list
          do
          rm --force "$file"
          ret=$?
          if [ $ret -ne 0 ]
              then
              report "PROBLEM" "rm returned $ret for file: $file"
              cleanexit=$E_RM_FAIL      # reset script return
          fi
          let "rmcount++"
        done

        # create reporting string for final_line

        rmplural=""
        test $rmcount -gt 1 && rmplural="s"  # plural form
        printf -v cleanstr " %d file%s deleted " $rmcount "$rmplural"

        controlled_exit $cleanexit
        ;;
esac
```

```
# ---------------------------------
#  set pipefail
# ---------------------------------

# TOFIX: pipefail: test and either comment full or
# remove not unequivocal as grep returns 1 on no match,
# but might help
#
# causes a pipeline to return the exit status of the
# last command in the pipe that returned a non-zero
# return value

set -o pipefail  # was shopt with -q to suppress output
                 # report

# seems useful where no boost is concerned

# -------------------------
#  report date and svn
# -------------------------

date=$( date '+%a %d-%b-%Y %H:%M:%S %Z %z' )
report "date" "$date"
report "svn" "$svnver"
calldir=$( pwd -P )
calldir=${calldir#/home/robbie/synk/xeona/}  # prune for readability
report "current branch" "$calldir"
report "command-line" "\$ $SCRIPT $cline"
test "$svn_flag" == "stale" && report "CAUTION" "svn $mess needed"

# -------------------------
#  confirm makefile
# -------------------------

makefile=$( ls -1 [Mm]akefile 2>/dev/null )
makefile=$( echo $makefile )            # strip newlines and surplus white-space

if [ -z "$makefile" ]
    then
    report
    report "ERROR" "no makefile found: [Mm]akefile"
    report
    exit $E_MISSING_FILES
fi

report "makefile name" "$makefile"

# ---------------------------------
#  test_simple_name()
# ---------------------------------

function test_simple_name
{
    local NAME="$1"
    local name="$2"

    test "$NAME" == "$name" || report "PROBLEM" "name mismatch assumption: $NAME != $name"
}

# ---------------------------------
#  source the unit-test lists
# ---------------------------------

# CAUTION: unit-test lists: these are no longer kept in
# this file for reasons of maintainability -- see
# instead "UNIT_TEST_LISTS".  Note also the execute
# permissions on the "source" file need not be set.

case "$type" in
    unit)
        utsource1="./$UNIT_TEST_LOCAL"       # concatenate to produce filename

        script=$( which "$SCRIPT" )          # script path in full
        path=$( dirname $script )            # directory part without trailing slash
        utsource2="$path/$UNIT_TEST_LISTS"   # concatenate to produce filename

        # hunt for local then universal variants

        if   [ -f "$utsource1" ]             # regular file found
            then
```

```
            utsource="$utsource1"            # local variant
        elif [ -f "$utsource2" ]             # regular file found
            then
            utsource="$utsource2"            # universal variant
        else
            report "PROBLEM" "unit-test lists not found, tried: $utsource1 $utsource2"
        fi

        # process that file

        if   [ ! -s "$utsource" ]            # regular file found but zero bytes
            then
            report "PROBLEM" "unit-test lists found but zero-bytes: $utsource"
        else                                 # success
            report "unit-test lists" "$utsource"
            source "$utsource"               # bash 'source' command, same as "."
        fi

        # remove multiple blanks

        SOURCES=$( echo $SOURCES )           # cheap way of stripping surplus white-space, beware ⬋
leading -neE

        # check sources list

        test -z "$SOURCES" && report "WARNING" "empty sources list"
        ;;
esac

# ---------------------------------
#  find_role()
# ---------------------------------

# CAUTION: the UNIX utility "expr" is NOT intuitive -- pay
# attention to the following:
#
#     "expr match" attempts to match the 'regex' pattern,
#     looks from the BEGINNING of host string, and
#     returns the number of matching characters
#
#     "expr match \(\)" is similar but attempts to extract
#     the 'regex' match, it also looks from the BEGINNING
#     of the host string
#
#     "express index" reports the one-based position of
#     the first CHARACTER in the substring list that
#     hits
#
#     strong quotes (') for the 'regex' or char-list are
#     suggested by Cooper (2007) (Advanced bash-scripting
#     guide), but weak quotes (") also seem to work fine
#     in the following contexts

function find_role
{
    local unittest=""                    # unit test file

    # hunt for the "*.ut[0-9]*" file, stated in terms of glob syntax
    for file in $SOURCES
      do
      test $( expr match "$file" '.*\.ut[[:digit:]]' ) -ne 0 && unittest="$file"
    done

    # now grab the value, the '/' demarcate a 'regex' which identifies the "file-role" key
    role=$( gawk 'BEGIN { FS = " : " } / file-role *:/ { print $2 }' "$unittest" )   # could also use ⬋
'awk'

    # strip any trailing " / unit test" and similar text
    role=${role%/*}                                  # strip final (or only) trailing "/" and ⬋
remainder

    # strip any trailing 'space' and 'tab' chars, note the "complemented character alternative" char ⬋
'^'
    role=$( expr match "$role" '\(.*[^[:space:]]\)' )   # applies to beginning of string
}

# ---------------------------------
#  find_level()
# ---------------------------------

# find digit in "name.ut[1-9].cc" from SOURCES
```

```
function find_level
{
    local file=""
    for file in $SOURCES                # CAUTION: no soft-quotes on 'SOURCES'
      do
      file="${file%.cc}"                # strip trailing ".cc" as 'bash' parameter substitution
      level="${file#[[:alpha:]]*\.ut}"  # strip leading part "abcde.ut" as 'regex'
      case "$level" in
          [0-9]) break ;;               # break on single digit
          *) level="(not found)" ;;     # should not get here
      esac
    done
}

# ---------------------------------
#  get role and level
# ---------------------------------

case "$type" in
    app)
        :
        ;;
    unit)
        find_role                       # try to obtain "file-role" value
        find_level                      # obtain test level 0 thru 9, only reported for units
        ;;
esac

# ---------------------------------
#  report output and sources
# ---------------------------------

case "$type" in
    app)
        report "output name" "$BINARY"
        report "make source names"      # can add a second "" for :
        make ccs                        # NOTE: as reported by make
        ;;
    unit)
        report "unit level"    "$level"
        report "unit role "    "$role"
        report "output name"   "$BINARY"
        report "source names"  "$SOURCES"
        report
        ;;
esac

# -------------------------
#  make_ret_test()
# -------------------------

function make_ret_test
{
    make_ret=$?                         # CAUTION: must be first line

    report
    if [ $make_ret -ne 0 ]
        then
        test $test_flag -eq 0 && rm --force "$maktemp" # file no longer needed
        local meaning=""
        case "$make_ret" in
            0)      meaning="success"          ;;      # not strictly required
            $CTRLC) meaning="probable ctrl-c"  ;;
            *)      meaning="fail"             ;;
        esac
        report "warning" "compiler exit $make_ret ($meaning)"
        let "xstat += $X_COMPILE_FAIL"
        controlled_exit $xstat
    fi
    return 0
}

# -------------------------
#  extra screening
# -------------------------

# check settings in relation to release builds

release_error_flag=0
update_error_flag=0
```

```
rstring="option -r (release build)"

case "$release_flag" in
    1)
        case "$svn_flag" in
            stale)
                string="must svn commit and update prior to $rstring"
                report "ERROR" "$string"
                update_error_flag=1
                ;;
        esac
        case "$debug_omit_flag" in
            0)
                string="must use option -d (omit _XDEBUG macro) with $rstring"
                report "ERROR" "$string"
                release_error_flag=1
                ;;
        esac
        case "$fresh_flag" in
            0)
                string="must use option -f (fresh compile) with $rstring"
                report "ERROR" "$string"
                release_error_flag=1
                ;;
        esac
        ;;
esac

# override the usual protections for release builds,
# only for developing scripts and such

override=0
case "$override" in
    1)
        echo "** $SCRIPT: CAUTION: override in force, resetting error flags: $override **"
        update_error_flag=0
        release_error_flag=0
        echo
        ;;
esac
# end of override code

test $update_error_flag  -ne 0 && controlled_exit $E_UPDATE_ISSUES
test $release_error_flag -ne 0 && controlled_exit $E_RELEASE_ISSUES

# ---------------------------------
#  process make options
# ---------------------------------

mops=""                                 # make options
case "$fourcore_flag" in
    1) mops="--jobs=4" ;;
esac

# -------------------------
#  process compiler options
# -------------------------

# at the time of writing, few if any assumptions were
# made about the actual compiler flag settings in the
# project makefile

cpp=""                                  # makefile variable : CPPFLAGS
cxx=""                                  # makefile variable : CXXFLAGS
ldd=""                                  # makefile variable : LDFLAGS
bin=""                                  # makefile variable : binary
src=""                                  # makefile variable : sources
targets=""

# cpp : 'CPPFLAGS=' preprocessor flags

node=$( uname --nodename )
case "$node" in
    sojus) termcols="-D_XTCOLS=142"              ;;   # [1]
    gogol) termcols="-D_XTCOLS=171"              ;;   # [2]
    hinau) termcols="-D_XTCOLS=166"              ;;   # [3]
    *)     termcols="-D_XTCOLS=122"              ;;   # recommended minimum
esac

# _XTCOLS sets the compiled-in terminal output width -- the recommened minimum being 122
```

```
# to check  : $ stty --all # refer to columns value OR $ stty size # take second value
# to obtain : cols = $( stty size | awk '{ print $2 }' )
# to test   : $ for i in $( seq 150 ); do printf "%${i}d\n" $i; done
#
# [1] 'sojus' with a small font, Gnome full screen (f11), and scroll bar, supports 143 chars
#             hardware Toshiba Tecra A2 330 : 1024 x 768
# [2] 'gogol' with a small font, Gnome full screen (fll), and scroll bar, supports 172 chars
#             hardware Toshiba Satellite Pro 6100 : 1400 x 1050
# [2] 'hinau' with a normal font, Gnome full screen (fll), and scroll bar, supports 167 chars
#             hardware Toshiba Tecra A11 11H : 1366x768

case "$debug_omit_flag" in
    0) debug="-D_XDEBUG"                          ;;
    1) debug=""                                   ;;   # override makefile default of -D_XDEBUG
esac

case "$svn_flag" in
    clean) svv="-D_XSVNREV=$svnver"               ;;   # update whenever acceptable
    *)     svv=""                                 ;;
esac

case "$release_flag" in
    0)
        release=""
        ;;
    1)                                                 # release build signaled
        release="-D_XRELEASE -DNDEBUG"                 # NDEBUG is official
        ;;
esac

case "$CPPMACRO" in
    "") macro=""                                  ;;
    *)  macro="-D_$CPPMACRO"                      ;;
esac

case "$type" in
    unit) unittest="-D_XUTEST"                    ;;   # unit test detected
    *)    unittest=""                             ;;
esac

case "$utest_omit_flag" in
    0)                                            ;;
    1) unittest=""                                ;;   # override default behavior
esac

case "$filesys_flag" in
    0) filesys=""                                 ;;
    1) filesys="-DBOOST_FILESYSTEM_NO_DEPRECATED" ;;
esac

case "$filesys_v3_flag" in
    0) filesys_v3=""                              ;;
    1) filesys_v3="-DBOOST_FILESYSTEM_VERSION=3"  ;;
esac

cppbuff="$termcols $debug $unittest $svv $release $macro $filesys $filesys_v3"
cppbuff=$( echo $cppbuff )

case "$cppbuff" in
    "") cpp="CPPFLAGS="                           ;;
    *)  cpp="CPPFLAGS=\"$cppbuff\""               ;;
esac

# cxx : 'CXXFLAGS=' compiler flags

case "$dryrun_flag" in
    0) dryrun=""                                  ;;
    1) dryrun="-###"                              ;;
esac

case "$quiet_flag" in
    0) quiet="-Wall"                              ;;
    1) quiet=""                                   ;;
esac

case "$release_flag" in
    0)
        release="$CXXDEBUG -O0 -fno-inline"            # 'CXXDEBUG' debug options set in preamble
        case "$profile_flag" in
            1) release="$release $CXXLDPROF"      ;;   # 'CXXLDPROF' profiling option set in preamble
        esac
```

```
        ;;
    1)                                                  # release build signaled
        release="-O3 -fno-strict-aliasing"
        ;;
esac

screenbuf=""

# note the new screen for the boost headers, which was:
# | grep --invert-match '/usr/local/include/boost-1_34_1/boost'"

case "$pedantic_flag" in
    0) pedantic=""                                  ;;
    1)
        pedantic="-pedantic"                # acceptable for Boost 1.34.1
        pedantic="$pedantic -Wno-long-long"  # added for Boost 1.35 (but useful earlier)
        screenbuf="\
| grep --invert-match '/usr/local/include/boost'"
        ;;
esac

case "$permissive_flag" in
    0) permissive=""                                ;;
    1) permissive="-fpermissive"                    ;;
esac

case "$no_gnu_flag" in
    0) ansi=""                                      ;;
    1) ansi="-std=c++98"                            ;;
esac

# the pedantic screen is a subset of the meyers screen so
# that it can just be simply overwritten and not merged

case "$meyers_flag" in
    0) meyers=""                                    ;;
    1)
        meyers="-Weffc++"

        # this filter variously used 'std::tr1::enable_shared_from_this'
        # and 'boost::enable_shared_from_this' but now just the common
        # pattern '::enable_shared_from_this'

        screenbuf="\
| grep --invert-match '/usr/local/include/boost' \
| grep --invert-match '/usr/lib/gcc' \
| grep --invert-match '::enable_shared_from_this' \
| grep --invert-match 'boost::noncopyable' \
| grep --invert-match 'boost::static_visitor'"

        # extend the screen if the lessfilter flag is NOT set

        case "$lessfilter_flag" in
            0)
                screenbuf="$screenbuf \
| grep --invert-match 'instantiated from' \
| grep --invert-match 'At global scope' \
| grep --invert-match '.. .... ........ from '"   # "In file included from "
                ;;
        esac
        ;;
esac

# simple screen based on namespaces
#       screenbuf="\
# | grep --invert-match 'boost::' \
# | grep --invert-match 'std::tr1::'"
#
# screen based on namespaces, entities, and messages (the last is for -pedantic)
#       screenbuf="\
# | grep --invert-match 'boost::' \
# | grep --invert-match 'std::tr1::' \
# | grep --invert-match 'mpl_::' \
# | grep --invert-match 'std::locale::_M_impl' \
# | grep --invert-match 'has a non-virtual destructor' \
# | grep --invert-match 'use of C99 long long integer constant'"
#
# screen based on file paths mostly
#       (see above)

cxxbuff="$dryrun $quiet $release $permissive $pedantic $ansi $meyers"
```

```
cxxbuff=$( echo $cxxbuff )

case "$cxxbuff" in
    "") cxx="CXXFLAGS="                 ;;
    *)  cxx="CXXFLAGS=\"$cxxbuff\""     ;;
esac

# ldd : 'LDFLAGS=' linker flags

# GLPK run-time library hack
#
#  LIBDIR="/usr/local/lib"
#  glpk430="-Wl,-rpath -Wl,$LIBDIR" # note '--rpath' also works

glpk430=""                              # empty

case "$dryrun_flag" in
    0) dryrun=""                        ;;
    1) dryrun="-###"                    ;;
esac

case "$release_flag" in
    0)
        case "$profile_flag" in
            0) profile=""               ;;
            1) profile="$CXXLDPROF"     ;;   # option must be same as the compile calls
        esac
        ;;
    1) profile=""                       ;;
esac

# static linking in general: see Stephens etal (2006
# sec1.23 p92-95) on specifying run-time library
# variants and also (p92) on the Boost libraries option
# "runtime-link=static" (but confirm currency)

# STATIC LINKING AND VALGRIND: the use of '-static'
# plus "valgrind" causes an avalanche of "Conditional
# jump or move depends on uninitialised value(s)"
# errors -- in the order of 80,000 or perhaps double,
# it seems, if more optimizations are applied.  The
# first message is as follows (edited, the xeona paths
# were full) -- it starts well before any xeona code:
#
#    Conditional jump or move depends on uninitialised value(s)
#       at 0x85639F5: __register_atfork (in ./xeona.r4287)
#       by 0x854F331: ptmalloc_init (in ./xeona.r4287)
#       by 0x8552AA5: malloc_hook_ini (in ./xeona.r4287)
#       by 0x8552307: malloc (in ./xeona.r4287)
#       by 0x8580E3B: _dl_init_paths (in ./xeona.r4287)
#       by 0x85641CB: _dl_non_dynamic_init (in ./xeona.r4287)
#       by 0x85648B5: __libc_init_first (in ./xeona.r4287)
#       by 0x853599D: (below main) (in ./xeona.r4287)
#
# No memory leaks occur.  A search of the 3.5.0
# documentation using "-static" revealed nothing (3.2.0
# is installed). Removing '-static' fixed the problem
# completely.  Note that the use of '-static' causes
# 'makefile' to link to '-lpthread'.
#
# Closure: I provisionally concluded this was not an
# application programming issue and decided to ignore
# 'valgrind' on this occasion.

case "$release_flag" in
    0)
        static=""                       # dynamic linking where possible
        ;;
    1)
        static=""                       # dynamic linking where possible
        static="-static"                # static linking in every case (see story above)
        ;;
esac

lddbuff="$glpk430 $dryrun $profile $static"
lddbuff=$( echo $lddbuff )

case "$lddbuff" in
    "") ldd="LDFLAGS="                  ;;
    *)  ldd="LDFLAGS=\"$lddbuff\""      ;;
esac
```

```
# bin : 'binary=' filename

bin="binary=\"$BINARY\""                # no manipulation required

# src : 'source=' list of sources

case "$type" in
    app)
        src=""                              # use full sources list as per makefile
        sources="(using makefile default)"  # for reporting purposes
        ;;
    unit)
        src="sources=\"$SOURCES\""      # use unit-test sources list as defined here
        sources=$SOURCES                # for reporting purposes
        ;;
esac

# targets : make target list, which can include controls like cleaning directories

case "$fresh_flag" in
    1) targets="$targets clean"         ;;
esac

       targets="$targets all"

case "$klean_flag" in
    1) targets="$targets tidy"          ;;
esac

targets=$( echo $targets )

case "$nofilter_flag" in
    0) screen="2>&1 $screenbuf"         ;;  # add leading redirection
    1) screen=""                        ;;  # nullify (after possibly setting earlier)
esac

# -------------------------
#  report environment
# -------------------------

# process header and library search path information

sepn=""
inc_search=""
lib_search=""
test -n "$CPATH"        && inc_search="CPATH=\"$CPATH\""
test -n "$LIBRARY_PATH" && lib_search="LIBRARY_PATH=\"$LIBRARY_PATH\""
test -n "$CPATH" -a -n "$LIBRARY_PATH" && sepn="  "
gccenv="$inc_search$sepn$lib_search"

report "shell environment" "${gccenv:-(not set)}"

# ---------------------------------
#  process special makefile name
# ---------------------------------

case "$SPECIAL_MAKEFILE" in
    "")                                 # SPECIAL_MAKEFILE not set
        spmakf=""
        ;;
    *)
        spmakf="--file=\"$SPECIAL_MAKEFILE\""
        ;;
esac

report "special makefile" "${SPECIAL_MAKEFILE:-(not set)}"

# -------------------------
#  report make options
# -------------------------

report "make options"       "${mops:-(none)}"
report "preprocessor flags" "${cppbuff:-(none)}"
report "compiler flags"     "${cxxbuff:-(none)}"
report "linker flags"       "${lddbuff:-(none)}"
report "sources"            "$sources"
report "output name"        "$BINARY"
report "make targets"       "$targets"
report "output screen"      "${screen:-(no screen)}"
```

```
# ---------------------------------
#  testfile()
# ---------------------------------

testfile_error_flag=0

function testfile
{
    test -f "$1" && return 0
    testfile_error_flag=1
    return 1
}

# ---------------------------------
#  check for unit-test files
# ---------------------------------

report

case "$type" in
    unit)
        for file in $SOURCES
          do
          testfile "$file" || report "ERROR" "file not found: $file"
        done
        ;;
esac

case "$testfile_error_flag" in
    1)
        report "ERROR" "quiting without action"
        controlled_exit $E_MISSING_FILES
        ;;
esac

# ---------------------------------
#  common.cc alignment
# ---------------------------------

# this code ensures that the xeona::DBUG variable, in
# turn set by the _XDEBUG preprocessor macro, will
# propagate properly through the app at link-time (this
# is also enforced by a touch command contained in the
# makefile) [prior to r1238 this code trivially
# modified 'common.cc' to give update the revision date
# as well]

# touch file and report
touch "$COMMON.cc"                      # also undertaken in makefile
report "$COMMON.cc" "file touched to ensure current xeona::DBUG"

# ---------------------------------
#  refresh xedoc file
# ---------------------------------

case "$type" in
    app)
        call_xedoc "refresh"            # function call, also reports
        ;;
esac

# -------------------------
#  compiler calls
# -------------------------

# CAUTION: the LDLIBS are defined in the makefile and
# no attempt is made to influence their setting here

for target in $targets
  do
  case "$target" in
      tidy|clean)
          call="make $spmakf $bin $target"
          call=$( echo $call )          # strip newlines and surplus white-space
          report "make call" "$call"
          report
          eval "$call"
          make_ret_test
          ;;
      all)
          call="make $spmakf $mops $cpp $cxx $ldd $bin $src $target $screen"
```

```
          call=$( echo $call )          # strip newlines and surplus white-space
          callreport=$call              # to be copied into SVNVERFILE
          report "make call" "$call"
          report
          maktemp="$TEMPSTUB-mak.txt"   # safe temporary filename
          eval "$call 2>&1 | tee $maktemp"
          make_ret_test                 # control branches here under error

          grep --quiet "warning" "$maktemp"       # NOTE: search term
          makgrepret=$?
          test $test_flag -eq 0 && rm --force "$maktemp"    # no longer needed
          case "$makgrepret" in
              0) gccwarn=1;  makmsg="compiler warnings issued" ;;
              1) gccwarn=0;  makmsg="no compile warnings"      ;;
              *) gccwarn=99; makmsg="unknown grep return"      ;;
          esac

          test $gccwarn -eq 1 && let "xstat += $X_COMPILE_WARN"

          gccrep="grep signal $gccwarn ($makmsg)"
          report "gcc status" "$gccrep"
          report

          # make binary read-only
          test -f $BINARY && chmod a-w $BINARY
          ;;
      *)
          report "ERROR" "unsupported compiler target: $target"
          ;;
  esac

# -->  # TOFIX: incorporate pipeline mods or remove
#
#    # $screen creates a pipeline with grep
#    function retmet { return $1; }        # ensure correct return
#    screen = "'boost::'"                  # add earlier
#    if [ -n "$screen" ]
#        then
#        make "$cpp $cxx $bin $src $target" 2>&1 | grep --invert-match 'boost::'
#       #make "$cpp $cxx $bin $src $target" 2>&1 | grep --invert-match "$screen"
#        retmet ${PIPESTATUS[0]}           # must follow call chain, PIPESTATUS is built-in
#        make_ret_test
#    else
#        make "$cpp $cxx $bin $src $target"
#        make_ret_test
#    fi
#
# <--

done

# ---------------------------------
#  nullify xedoc file
# ---------------------------------

case "$type" in
    app) call_xedoc "nullify" ;;        # function call, also reports
esac

# ---------------------------------
#  write out run details
# ---------------------------------

# pause for a moment in some cases and perhaps also beep

#  let "sum = $pedantic_flag + $meyers_flag + $meyers_filter_flag"
#  if [ $sum -gt 0 ]

if [ $alert_flag -gt 0 ]
    then
    alert -1
    sleep  3
fi

# report

report "$SVNVERFILE" "overwriting file"

svnverfile=$( svnversion $SVN_BASE 2>/dev/null )

rm --force "$SVNVERFILE"
```

```
{
    echo
    echo "last build details under $SCRIPT (as opposed to make)"
    echo
    echo "binary    : $BINARY"
    echo "type      : $type"
    echo "script    : $SCRIPT $cline"
    echo "machcount : $countstr"
    echo "local-svn : $svnverfile"
    echo "date      : $date"
    echo "make-call : $callreport"
    echo
} >> "$SVNVERFILE"

test $test_flag -ne 0 && cp "$SVNVERFILE" "$TEMPSTUB-cal.txt"

# -------------------------
#  report binary info
# -------------------------

report "binary name" "$BINARY"

BYTES=$( stat -c %s "$BINARY" 2>/dev/null )  # -c %s is total size in bytes
if test -z $BYTES
    then
    report "binary size" "file not present"
    nobin=1
    controlled_exit $xstat
else
    report "binary size" "$BYTES bytes"
    if [ $BYTES -eq 0 ]
        then
        emptybin=1
        controlled_exit $xstat
    fi
    binperms="(stat call not made)"
    test -f "$BINARY" && binperms=$( stat --format=%A $BINARY )
    report "binary perms" "$binperms"
fi

# ---------------------------------
#  abandon execution as needed
# ---------------------------------

case "$no_execute_flag" in
    1)
        report "note" "option -x (do not run binary) is in effect"
        controlled_exit $xstat
        ;;
esac

# ---------------------------------
#  binexit_means()
# ---------------------------------

# the previous hardcoded values are in r2769

binexit_meaning=""

function binexit_means
{
    local binexit="$1"
    deport "binexit" "$binexit"

    case "$type" in
        app)
            # try to recover interpretation from xeona
            binexit_meaning=$( ./$BINARY --output $binexit 2>/dev/null )   # can be empty string
            deport "--output result" "$binexit_meaning"

            # overwrite special cases
            local abortmsg="probable terminate on uncaught throw OR glibc detects invalid pointer ⬋
during free call"
            case "$binexit" in
                0)         binexit_meaning="success"                      ;;    # shorter version
                $CTRLC)    binexit_meaning="possible ctrl-c"              ;;
                $ABORT)    binexit_meaning="$abortmsg"                    ;;
                $SEGFAULT) binexit_meaning="probable segmentation fault"  ;;
            esac
```

```
            # offer default message if above code fails
            test -z "$binexit_meaning" && binexit_meaning="undocumented application return"
            deport "binexit_meaning" "$binexit_meaning"
            ;;
        unit)
            # units are NOT typically code to return non-zero when run-time problems are encountered
            case "$binexit" in
                0) binexit_meaning="success"                              ;;
                1) binexit_meaning="uncharacteristic failure"             ;;
                *) binexit_meaning="undocumented unit test return"        ;;
            esac
            ;;
    esac
}

# ---------------------------------
#  run binary (unit or app)
# ---------------------------------

redirect=""

case "$logger_only_flag" in
    1) redirect="$redirect 1>/dev/null" ;;
esac

case "$screen_only_flag" in
    1) redirect="$redirect 2>/dev/null" ;;
esac

timing=""                               # null command
timing="time -p"                        # shell keyword (v3.1.17)
timing="/usr/bin/time --portability"    # GNU utility (v1.7)

lead="."                                # on binary name
test -d "$TESTSDIR" && lead=".."        # assume test subdirectory is just one deep

case "$type" in
    app)
        run="$timing $lead/$BINARY $APP_ARGS $redirect"
        report "type"              "application"
        report "application args" "$APP_ARGS"
        ;;
    unit)
        run="$timing $lead/$BINARY $redirect"
        report "type" "unit test"
        ;;
    *)
        :                               # dummy statement needed
        ;;
esac

run=$( echo $run )
report "run call" "$run"

echo
echo "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

test "$lead" = ".." && cd $TESTSDIR     # cd up as required

eval "$run"
runret=$?                               # grab exit status

test "$lead" = ".." && cd $lead         # cd down as required

echo "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
echo

binexit_means $runret                   # sets 'binexit_meaning'

binexit="$BINARY exit $runret ($binexit_meaning)"  # for later reporting
report "run status" "$binexit"

test $runret -ne 0 && let "xstat += $X_RUN_FAIL"
test $runret -ne 0 && report "warning" "run-time failure"

# ---------------------------------
#  valgrind binary
# ---------------------------------

# define basic valgrind options
#
```

```
#     --tool=memcheck     default in any case
#     --error-exitcode=1  return 1 on non-success (default of 0 returns process exit)

VALGRIND_OPTS0="--tool=memcheck --error-exitcode=1"

valcmd="$VALGRIND $VALGRIND_OPTS0"

case "$valgrind_extd_flag" in
    1)
        valgrind_flag=1                 # -v need not have been set
        valcmd="$VALGRIND $VALGRIND_OPTS0 $VALGRIND_OPTS1"
        ;;
esac

case "$valgrind_flag" in
    1)
        valtemp="$TEMPSTUB-val.txt"     # safe temporary filename
        report
        report "valgrind" "about to memory check binary"

        lead="."                                # on binary name
        test -d "$TESTSDIR" && lead=".."     # assume test subdirectory is just one deep

        case "$type" in
            app)
                valcall="$valcmd $lead/$BINARY $APP_ARGS"
                ;;
            unit)
                valcall="$valcmd $lead/$BINARY"
                ;;
            *)
                :                       # dummy statement needed
                ;;
        esac

        valcall=$( echo $valcall )
        report "valgrind call" "$valcall"
        report

        test "$lead" = ".." && cd $TESTSDIR  # cd up as required

        eval "$valcall 2>&1 | tee $lead/$valtemp" # CAUTION: stderr redirection is required
        valcmdexit=$?

        test "$lead" = ".." && cd $lead      # cd down as required

        case "$valcmdexit" in
            0)          valerror=0;  valerrormsg="no memory errors"                                    ⬋
;;
            1)          valerror=1;  valerrormsg="memory errors occurred"; let "xstat += ⬋
X_MEMORY_ERROR" ;;
            $CTRLC)     valerror=99; valerrormsg="possible ctrl-c"                                     ⬋
;;
            $SEGFAULT) valerror=99; valerrormsg="possible segmentation fault"                          ⬋
;;
            155)        valerror=99; valerrormsg="perhaps profile-enabled binary leaking memory (gmon)" ⬋
;;
            *)          valerror=99; valerrormsg="unknown valgrind return"                             ⬋
;;
        esac

        grep --quiet "LEAK" "$valtemp"
        valgrepret=$?
        test $test_flag -eq 0 && rm --force "$valtemp"
        case "$valgrepret" in
            0) valleak=1;  valleakmsg="memory leaks occurred"; let "xstat += X_MEMORY_LEAK" ;;
            1) valleak=0;  valleakmsg="no memory leaks"                                     ;;
            *) valleak=99; valleakmsg="unknown grep return"                                 ;;
        esac

        report
        report "gcc status (again)" "$gccrep"     # repeat from earlier
        report "run status (again)" "$binexit"    # repeat from earlier
        report "memory status"      "valgrind exit $valcmdexit ($valerrormsg)"
        report "leak status"        "grep signal $valleak ($valleakmsg)"

        let "valstat = $valerror + $valleak"
        ;;
    0)
        :
        ;;
```

```
esac

# ---------------------------------
# xedocs_call()
# ---------------------------------

function xedocs_call
{
    local binary="./$BINARY"                # for convenience
    local subdir="$TESTSDIR"                # for convenience

    local xemtemp="$subdir/$TEMPSTUB.xem"    # safe temporary filename
    local logtemp="$subdir/$TEMPSTUB.log"

    # create the xedocs model file
    {
        $binary --xem head
        echo "------------------------------------------- test entities"
        $binary --class .                   # note the dot
        $binary --xem tail
    } 2>/dev/null | cat --squeeze-blank > $xemtemp

    # run the model
    local call="$binary --report ${report:=2} --mode 4 --file $xemtemp"
    eval "$call > $logtemp 2>&1"
    ret=$?

    # interpret the return
    case $ret in
        0)
            report "xedocs suite" "run exit $ret (xedocs valid)"
            test $test_flag -eq 0 && rm --force "$xemtemp"  # file no longer needed
            test $test_flag -eq 0 && rm --force "$logtemp"  # file no longer needed
            ;;
        *)
            reason=$( $binary --output $ret 2>/dev/null )
            report
            report "xedocs suite" "run exit $ret = $reason"
            report "xedocs suite" "call: $call"
            report "xedocs suite" "allied files: $xemtemp $logtemp"
            report
            ;;
    esac

    return $ret
}

# ---------------------------------
#  run xedocs as required
# ---------------------------------

case "$xedocs_flag" in
    1)
        case "$type" in
            app)  xedocs_call || let "xstat += $X_XEDOCS_FAIL"                          ;;
            unit) deport "xedocs test" "option -e (ensure) not relevant for a unit test"  ;;
        esac
        ;;
esac

# ---------------------------------
#  xmok_call()
# ---------------------------------

function xmok_call
{
    local xmktemp="$TEMPSTUB-xmk.txt"   # safe temporary filename

    local call="$XMOK $XMOK_ARGS ./$BINARY"
    eval "$call > $xmktemp 2>&1"
    local ret=$?

    function xmok_report
    {
        echo
        cat "$xmktemp" | grep "^$XMOK:   call "
        echo
        $XMOK --aggret                  # report sought aggregate return
    }

    case $ret in
```

```
        0)
            report "xem tests" "xmok exit $ret (test suite passed)"
            ;;
        1)
            report "xem tests" "xmok exit $ret (test suite issues)"
            xmok_report
            ;;
        *)
            report "xem tests" "xmok exit $ret (xmok failure)"
            ;;
    esac

    test $test_flag -eq 0 && rm --force "$xmktemp"     # no longer needed

    return $ret
}

# ---------------------------------
#  call xmok as required
# ---------------------------------

case "$xmok_flag" in
    1)
        case "$type" in
            app)  xmok_call || let "xstat += $X_XMOK_FAIL"                          ;;
            unit) deport "xem tests" "option -o (okay) not relevant for a unit test"  ;;
        esac
        ;;
esac

# -------------------------
#  delete binary as needed
# -------------------------

case "$Klean_flag" in
    1) delete_binary  ;;
esac

# -------------------------
#  housekeeping
# -------------------------

controlled_exit $xstat

report
report "CODING ERROR" "at end of script, perhaps controlled_exit() is faulty"
report

exit 255                                # should never get here

# ---------------------------------
#  junk code
# ---------------------------------

# printenv CPATH | gawk 'BEGIN { RS = ":" } { if ($0 == "/usr/local/include/boost-1_38") exit 1 }' && ⬋
export CPATH="$CPATH:$cpath"

#  end of file
```

```
#! /bin/bash

#  file-purpose     : run data tests (as opposed to unit tests)
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Tue 12-Aug-2008 20:10 UTC
#  file-status      : working
#  file-keywords    : xeona

#  $Revision: 4806 $
#  $Date: 2010-07-20 13:38:14 +0200 (Tue, 20 Jul 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/scripts/xmok $

#  note: XEM means "xeona model"
#
#  parameter substitution
#
#   :- means use default value if parameter unset or empty
#   := means assign default value if parameter unset or empty and use that too

# ---------------------------------
#  settings
# ---------------------------------

XMOK_AGGREG_FILE="xeona-xmoks/xaggreg"  # contains 'AGGREG' the desired aggregate result
DEFAULT_BINARY="xeona.mach"             # binary name only, command-line options not accepted here

# ---------------------------------
#  preamble
# ---------------------------------

XEM="xem"                               # XEM extension
GUARD="guard"                           # XEM guard tag
MANDATORY_OPTIONS="--guard"

SDIR="xeona-xmoks"                      # subdirectory for XEM files
MAINCC="main.cc"                        # name of main file, used to confirm base directory

FILTER="2> /dev/null"
VALEEC=100
VALGRIND="valgrind --tool=memcheck --error-exitcode=$VALEEC --leak-check=full --show-reachable=yes"

SCRIPT=$(basename "$0")                 # this script

E_SUCCESS=0
E_FAILURE=1
E_USAGE=2
E_CONTEXT_FAILURE=64                    # errors or 64 or greater provoke a "FATAL" exit

# ---------------------------------
#  display_help()
# ---------------------------------

function display_help
{
local str1="else use './$DEFAULT_BINARY'"
local str2="space-separated list"

echo
echo "        usage: $SCRIPT <option>                                        test all with appropriate ⬋
defaults"
echo "               $SCRIPT <option> <models>                               test $str2 with ⬋
appropriate defaults"
echo "               $SCRIPT <option> <binary> [--binopt [val]] [models]     test using given binary ⬋
and binopt list"
echo "               $SCRIPT --aggret                           display sought aggregate return and ⬋
exit"
echo "               $SCRIPT --help                             display this message and exit"
echo "       option:      --              run tests using all defaults"
echo "      (single)      -b  --bucket    redirect 'stderr' to '/dev/null' (the bit bucket)"
echo "                    -0  --exittrip=0  use exit trip {1 2 3 4 5 6} on all files (ignore xem file ⬋
setting), implies -b"
echo "                    -x  --expunge   remove output files (default is to keep)"
echo "                    -r  --release   use highest 'xeona.r0000' binary in current *branch* ⬋
($str1)"
echo "                    -v  --valgrind  apply valgrind more (a memory error exits $VALEEC)"
echo "                    -y  --dry       print commands but no further action (dry run)"
echo "      purpose: run data tests using \"guard\" xems in hardcoded subdirectory '$SDIR'"
echo "       filter: any generated filenames which contain the regex \"_\" are automatically excluded"
echo "     requires: file '$MAINCC' be present in the current directory"
echo "     ordering: all       : process oldest to newest based on last-modified timestamp"
```

```
echo "               nominated : in order given, also the '.guard' qualifier may be omitted and ⬋
'$SDIR/' can be prepended"
echo "  model names: the extension '.$XEM' must be present"
echo "               the '.guard' qualifier may be omitted"
echo "               the path '$SDIR/' can be prepended"
echo "     examples: default run      : \$ $SCRIPT --"
echo "               added monitoring : \$ $SCRIPT -x xeona.r100 --report 1 --watch b/"
echo "               custom           : \$ $SCRIPT -- xeona.make one.xem two.xem"
echo "  bugs/issues: only one script option [-bkrvy] can be used at one time"
echo "               no script-level checks on \"--binopt val\" correctness"
echo "               valgrind memory leaks are NOT completion reported (although memory errors are)"
echo "   xem fields: the following \"run-script-settings\" fields are supported"
echo "               script-run-me            : 0 = skip, anything else = run"
echo "               script-option-exittrip   : set --exittrip to given integer"
echo "               script-option-nodata     : set --nodata given non-zero argument"
echo "               script-option-jumpy      : set --jumpy given non-zero argument"
echo "    hardcodes: xem subdirectory         : $SDIR"
echo "               guard file pattern       : $SDIR/*.$GUARD.$XEM"
echo "               default binary           : ./$DEFAULT_BINARY"
echo "               mandatory binary options : $MANDATORY_OPTIONS"
echo "               valgrind call            : $VALGRIND"
echo

return 0
}

# ---------------------------------
#  highest_release_binary()
# ---------------------------------

# for 'find', both work, the second removes the "./"
# %p  file's name
# %P  file's name with the name of the command line argument under which it was found removed

function highest_release_binary
{
    # CAUTION: note the outer "()" on the 'glob' line
    local glob
    local bin
    glob=($(find -P . -maxdepth 2 -name "xeona.r[0-9][0-9][0-9][0-9]" -type f -printf "%p\n" | sort ⬋
--reverse))
    bin=${glob[0]}                      # grab highest release number
    bin=${bin#./}                       # strip any leading "./"
    test -n "$bin" && DEFAULT_BINARY="$bin"
}

# ---------------------------------
#  screen command-line
# ---------------------------------

commandline="$*"

filter=""
valgrind=""

case "$1" in
    --help|--hel|--he|--h|-help|-hel|-he|-h|"-?")
        display_help
        exit $E_SUCCESS
        ;;

    --aggret|-a)
        source "$XMOK_AGGREG_FILE" ||
        {
            echo "$SCRIPT: WARNING: xmok aggregate file not sourced: $XMOK_AGGREG_FILE"
            exit $E_CONTEXT_FAILURE
        }
        echo "$SCRIPT: sought aggregate return: $AGGRET"
        exit $E_SUCCESS
        ;;

    --dry|-y)
        shift
        mode="dry"
        output="keep"                   # no files should be generated either
        ;;
    --expunge|-x)
        shift
        mode="run"
        output="remove"
        ;;
```

```
    --release|-r)
        highest_release_binary          # overwrites DEFAULT_BINARY
        shift
        mode="run"
        output="keep"
        ;;
    --bucket|-b)
        shift
        mode="filter"
        output="keep"
        ;;
    --valgrind|-v)
        shift
        mode="valgrind"
        output="keep"
        ;;
    --exittrip=[1-6]|-[1-6])
        opt="$1"                        # option as above
        shift
        mode="filter"
        output="keep"
        len=${#opt}                     # length of 'opt'
        let "len--"
        overtrip="${opt:$len}"          # obtain the last char
        echo "$SCRIPT: overtrip: $overtrip"
        ;;
    --exittrip0|-0)
        echo "$SCRIPT: this option requires {1 2 3 4 5 6}"
        exit $E_USAGE
        ;;
    "--")
        shift
        mode="run"
        output="keep"
        ;;
    -d)
        echo "$SCRIPT: unsupported option (you might want -y for dryrun)"
        exit $E_USAGE
        ;;
    "")
        echo "$SCRIPT: incorrect usage (try --help or perhaps --)"
        exit $E_USAGE
        ;;
    *)
        echo "$SCRIPT: unsupported option (try --help): $1"
        exit $E_USAGE
        ;;
esac

#  uncomment following two lines during development
#  echo "$SCRIPT: STRONG WARNING: returning mode to 'dry' for development purposes"
#  mode="dry"

# ---------------------------------
#  process command-line arguments
# ---------------------------------

# note zero-based indexing for strings

BINARY=""
OPTIONS=""
OTHER=""

loops=0
opt="false"
bin="notfound"
for arg in "$@"                         # CAUTION: use double quotes to retain command-line quoting
do
  let "++loops"
  if   [ $(basename -- "$arg") != $(basename -- "$arg" ".$XEM") ]     # CAUTION: note the "--" ⬋
protection
      then
      stem=$(basename "$arg" ".$XEM")
      stem=$(basename "$stem" ".$GUARD")     # strip if required
      model="$SDIR/$stem.$GUARD.$XEM"        # reassemble
      MODELS="$MODELS $model"
  elif [ ${arg:0:2} == "--" ]           # a leading -- indicates an option name
      then
      OPTIONS="$OPTIONS $arg"           # concatenate
      opt="true"
  elif [ "$opt" == "true" ]             # an option value
```

```
      then
      test $(expr index "$arg" " ") -ne 0 && arg="\"$arg\"" # contains space, therefore add back quotes
      OPTIONS="$OPTIONS $arg"           # concatenate
      opt="false"                       # toggle flag
  elif [ "$bin" == "notfound" ]
      then
      BINARY="$arg"                     # overwrite
      bin="found"                       # toggle flag
  else
      OTHER="$OTHER $arg"
  fi
done

OPTIONS=${OPTIONS#" "}                  # strip any leading space
MODELS=${MODELS#" "}                    # strip any leading space
OTHER=${OTHER#" "}                      # strip any leading space

supplied_binary="${BINARY:-(none)}"
supplied_options="${OPTIONS:-(none)}"
supplied_models="${MODELS:-(none)}"
supplied_other="${OTHER:-(none)}"

echo
echo "$SCRIPT: supplied binary                      : $supplied_binary"
echo "$SCRIPT: supplied options                     : $supplied_options"
echo "$SCRIPT: supplied models (.guard optional)    : $supplied_models"
echo "$SCRIPT: supplied other (ignored)             : $supplied_other"
echo "$SCRIPT: processing loops                     : $loops"

test -z "$BINARY" && BINARY="$DEFAULT_BINARY"
OPTIONS="$OPTIONS $MANDATORY_OPTIONS"
OPTIONS=${OPTIONS#" "}                  # strip any leading space

# ---------------------------------
#  define separator line
# ---------------------------------

# create line
char=":"                                # line character
len=$(stty size | gawk '{ print $2 }')  # dynamic setting
let "len = len - 2"
for i in $(seq $len); do line="$line$char"; done

# colorize line
case "$TERM" in
    dumb) line="$line"                                  ;;
    *)    line=$(echo -e "\033[01;35m$line\033[00m")  ;;    # 31=red 35=magenta
esac

# ---------------------------------
#  controlled_exit()
# ---------------------------------

function controlled_exit
{
    local exitcode="$1"
    local message="$2"

    if [ $exitcode -ge 64 ]
        then
        echo "$SCRIPT: $message"
        echo "$SCRIPT: FATAL: exiting with status: $exitcode"
        echo
    else
        echo
        echo "$SCRIPT: command line           : $SCRIPT $commandline"
        echo "$SCRIPT: elapsed time           : $SECONDS seconds"
        echo "$SCRIPT: run status             : $message"
        echo
    fi

    exit $exitcode
}

# ---------------------------------
#  confirm context
# ---------------------------------

fail_count=0                            # failure counter

if [ ! -f "$MAINCC" ]                   # file exists (not)
```

```
    then
    echo "$SCRIPT: context failure: file not found (as proxy for base directory): $MAINCC "
    let "fail_count++"
fi

if [ ! -d "$SDIR" ]                     #  file exists and is a directory (not)
    then
    echo "$SCRIPT: context failure: subdirectory not found: $SDIR"
    let "fail_count++"
fi

if [ ! -f "$BINARY" ]                   # file exists (not)
    then
    echo "$SCRIPT: context failure: nominated xeona binary not found: $BINARY (a model requires the ⬈
extension .$XEM)"
    let "fail_count++"
elif [ ! -x "$BINARY" ]                 # file exists and has execute permission granted (not)
    then
    echo "$SCRIPT: context failure: nominated xeona binary not executable: $BINARY"
    let "fail_count++"
fi

ls $SDIR/*.$GUARD.$XEM &> /dev/null     # redirection equivalent to "> /dev/null 2>&1"
ret=$?                                  # 0 = files found, 2 = "No such file or directory"
if [ $ret -ne 0 ]
    then
    echo "$SCRIPT: context failure: no XEM guard files found in subdirectory: $SDIR"
    let "fail_count++"
fi

test $fail_count -eq 0 || controlled_exit $E_CONTEXT_FAILURE "errors occurred: $fail_count"

# ----------------------------------
#  initial report
# ----------------------------------

echo
echo "$SCRIPT: mode     : $mode"
echo "$SCRIPT: binary   : $BINARY"
echo "$SCRIPT: options  : $OPTIONS"

# ----------------------------------
#  push_call()
# ----------------------------------

# used to store call-by-call results

declare -a rets
declare -a xems

aggret=0

function push_call
{
    local ret="$1"
    local xem="$2"
    local xop="$3"
    local size=${#rets[*]}
    let "++size"
    rets[$size]="$ret"
    xems[$size]="$xem"
    xops[$size]="$xop"
    let "aggret += ret"
}

# ----------------------------------
#  runsetting()
# ----------------------------------

# based on information in the 'program.run-script-settings' stanza

XOPTIONS=""
RUNME="no"

function runsetting
{
    local file="$1.$GUARD.$XEM"

    XOPTIONS=""
    RUNME="no"
```

```
    test -f "$file" || echo "$SCRIPT: PROBLEM: file not found: $xem" # added protection

    local       run_me=$(gawk 'BEGIN { FS = ">[[:blank:]]*" } /script-run-me/          { print $2 }' ⬋
"$file")
    local arg_exittrip=$(gawk 'BEGIN { FS = ">[[:blank:]]*" } /script-option-exittrip/ { print $2 }' ⬋
"$file")
    local   arg_nodata=$(gawk 'BEGIN { FS = ">[[:blank:]]*" } /script-option-nodata/   { print $2 }' ⬋
"$file")
    local    arg_jumpy=$(gawk 'BEGIN { FS = ">[[:blank:]]*" } /script-option-jumpy/    { print $2 }' ⬋
"$file")

    test -n "$overtrip" && arg_exittrip="$overtrip"

    local exittrip="--exittrip"
    local   nodata="--nodata"
    local    jumpy="--jumpy"

    local option_exittrip=${exittrip//[[:print:]]/' '}          # blank string of equivalent length
    local   option_nodata=${nodata//[[:print:]]/' '}            # blank string of equivalent length
    local    option_jumpy=${jumpy//[[:print:]]/' '}             # blank string of equivalent length

    option_exittrip="$option_exittrip  "                        # plus room for trailing argument

    test $arg_exittrip && option_exittrip="$exittrip $arg_exittrip"
    test   $arg_nodata && test $arg_nodata -ne 0 && option_nodata="$nodata"
    test    $arg_jumpy && test  $arg_jumpy -ne 0 &&  option_jumpy="$jumpy"

    XOPTIONS="$option_exittrip $option_nodata $option_jumpy"     # concatenate

    printf "$SCRIPT: runsetting: run-me = %s\n" "$run_me"
    printf "$SCRIPT: options = %s --file %s\n" "$XOPTIONS" "$file"

    test "$run_me" != "0" && RUNME="yes"     # string comparison
}

# ---------------------------------
#  main code
# ---------------------------------

echo

# finalize list of 'files'

files=""
noshows=""

if [ -n "$MODELS" ]                     # use given list
    then
    for file in $MODELS
      do
      if [ -f "$file" ]
          then
          files="$files $file"
      else
          noshow="$SCRIPT: input file not found: $file"
          printf -v noshows "%s\n%s" "$noshows" "$noshow"  # "leading" newline is correct
      fi
    done
    files=${files#" "}                  # strip any leading space
else                                    # generate list
    files=$(ls -rt $SDIR/*.$GUARD.$XEM) # CAUTION: no double quotes

    ## can exclude any generated filenames containing the regex "_" here
    ## files=$(echo "$files" | grep --invert-match "_")
fi

case "$mode" in
    dry)
        echo "$files"
        echo
        ;;
esac

# process list of 'files'

for xem in $files
do
  xem=${xem%.$GUARD.$XEM}               # strip trailing ".guard.xem"

  runsetting "$xem"                     # uses 'program.run-script-settings' stanza information
  case "$RUNME" in
```

```
      no)
          echo
          echo "$line"
          echo
          continue
          ;;
  esac
  test -n "$XOPTIONS" && LOPTIONS="$OPTIONS $XOPTIONS"

  case "$mode" in
      run)
          call="./$BINARY $LOPTIONS --file $xem"
          echo "$SCRIPT: call = $call"
          eval "$call"
          ret=$?
          test $ret -eq 11 && echo
          echo "$line"
          echo
          ;;
      valgrind)
          valgrind="$VALGRIND"
          call="$VALGRIND ./$BINARY $LOPTIONS --file $xem"
          echo "$SCRIPT: call = $call"
          echo
          eval "$call"
          ret=$?
          echo
          echo "$line"
          echo
          ;;
      filter)
          filter="$FILTER"
          call="./$BINARY $LOPTIONS --file $xem $filter"
          echo "$SCRIPT: call = $call"
          eval "$call"
          ret=$?
          echo "$line"
          echo
          ;;
      dry)
          echo "$SCRIPT: dry call : ./$BINARY $LOPTIONS --file $xem"
          ret=0
          ;;
      *)                                # should not be here
        echo "$SCRIPT: coding error: call loop"
        ;;
  esac
  push_call "$ret" "$xem" "${XOPTIONS#--}"   # may need to modify this last call
done

# ---------------------------------
#  report aggregate results
# ---------------------------------

test "$mode" == "dry" && echo

echo "$SCRIPT: final reporting"
echo

test "$mode" != "dry" &&
{
    echo "$SCRIPT: supplied binary                      : $supplied_binary"
    echo "$SCRIPT: supplied options                     : $supplied_options"
    echo "$SCRIPT: supplied models (.guard optional)    : $supplied_models"
    echo "$SCRIPT: supplied other (ignored)             : $supplied_other"
    echo "$SCRIPT: processing loops                     : $loops"
    echo
}

size=${#rets[*]}                        # number of XEM files processed

test $size -ge 1 &&
{
  # echo "$SCRIPT: mode     : $mode"
    echo "$SCRIPT: binary   : $BINARY"
    echo "$SCRIPT: options  : $OPTIONS"
    echo "$SCRIPT: filter   : ${filter:-(not set)}"
    echo "$SCRIPT: valgrind : ${valgrind:-(not set)}"
    echo
}
```

```
host=$(hostname --short)
case "$host" in
    hinau) WRAP=83  ;;                  # used to truncate 'exit_describe'
    sojus) WRAP=59  ;;
    *)     WRAP=59  ;;
esac

TAB=26                                  # alignment for XEM filenames (was 30)
printf "$SCRIPT: subdirectory : %s\n" "$SDIR/"
for i in $(seq $size)
  do
  # process xeona exit status
  exit_status=${rets[$i]}
  exit_describe=$( ./$BINARY --output $exit_status 2>/dev/null )
  test $exit_status -eq 0 && exit_describe="success"   # to simplify the message
  test -z "$exit_describe" && exit_describe="(exit status $exit_status problem)"
  exit_describe="${exit_describe:0:90}" # truncate if necessary (a length check may be necessary)
  # output
  file="${xems[$i]}"
  file="${file#$SDIR/}"
  exitsay="${exit_describe:0:$WRAP}"    # to prevent wrapping
  printf "$SCRIPT:   call %02d    %-*s   %12s  %3d = %s\n"  "$i" "$TAB" "$file" "${xops[$i]}" ↙
"${rets[$i]}" "$exitsay"
done

if [ $size -gt 1 ]
then
    echo
    let "tab = TAB + 19 + 30"           # somewhat adaptive
    printf "%-*s  %3d\n" $tab "$SCRIPT: aggregated returns :" $aggret
fi

# ---------------------------------
#  explain noshows
# ---------------------------------

test -n "$noshows" && echo "$noshows"

# ---------------------------------
#  removed generated files
# ---------------------------------

case "$output" in
    remove)
        echo
        count_output=0
        count_backup=0
        for xem in $files
          do
          xem=${xem%.$GUARD.$XEM}       # strip trailing ".guard.xem"
          test -f "$xem.$XEM"  && rm --force "$xem.$XEM"  && let "++count_output"
          test -f "$xem.$XEM~" && rm --force "$xem.$XEM~" && let "++count_backup"
        done
        echo "$SCRIPT: backup files removed: $count_backup"
        echo "$SCRIPT: output files removed: $count_output"
        ;;
    keep)
        echo
        count_backup=0
        for xem in $files
          do
          xem=${xem%.$GUARD.$XEM}       # strip trailing ".guard.xem"
          test -f "$xem.$XEM~" && rm --force "$xem.$XEM~" && let "++count_backup"
        done
        echo "$SCRIPT: backup files removed: $count_backup"
        echo "$SCRIPT: output files kept"
        ;;
    *)                                  # should not be here
        echo "$SCRIPT: coding error: remove loop"
        ;;
esac

# ---------------------------------
#  process aggregate return
# ---------------------------------

AGGRET=-1                               # nonsensical and clearly incorrect default value

source "$XMOK_AGGREG_FILE" || echo "$SCRIPT: WARNING: xmok aggregate file not sourced: ↙
$XMOK_AGGREG_FILE"
```

```
case $aggret in
    $AGGRET) controlled_exit $E_SUCCESS "successful completion"               ;;
    *)       controlled_exit $E_FAILURE "expected aggregate $AGGRET not met"  ;;
esac

# ---------------------------------
#  housekeeping
# ---------------------------------

exit 255                                # should not be here

#  $Id: xmok 4806 2010-07-20 11:38:14Z robbie $
#  end of file
```

```
#  file-purpose     : create breadboard plot
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Thu 04-Jun-2009 19:56 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 5089 $
#  $Date: 2010-08-16 19:51:55 +0200 (Mon, 16 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/breadplot.R $

#  OVERVIEW
#
#  associations  : tie / link / connect / couple / bind / span
#  relevant here : link to commodities and contexts / connect cables to sockets

# ---------------------------------
#  function : xem.createBreadset.1
# ---------------------------------
#  description  : takes recset and builds label-based conex
#  role         : breadplot data preparation
#  takes        : recset
#  returns      : conex1
#  techniques   : 'is.element', (asymmetric) 'setdiff', 'next'
#  status       : complete
# ---------------------------------

xem.createBreadset.1 <- function (recset)
{
  message("  info : xem.createBreadset.1 : commencing")
  if ( DBUG > 0 ) message()

  recordIds <- names(recset)                 # not used
  entityIds <- recset$metadata$ids           # collected as part of the build process

  if ( is.null(entityIds) ) stop("no entities specified in in model (malformed data)")

  breadset  <- list()                        # declare an empty list object
  p         <- 0                             # breadset counter

  for ( i in 1:length(entityIds) )           # loop the entities
    {
      if ( DBUG > 0 ) message("entity loop ", i, " with id ", entityIds[i])

      bread    <- list()                     # may or may not be inserted
      q        <- 0                          # bread counter

      entityId <- entityIds[i]
      entity   <- recset[[entityId]]         # grab the entity

      if ( entity$enabled == FALSE ) next    # skip disabled entities

      for ( j in 3:length(entity) )          # loop the fields, but skip the first two
        {
          if ( DBUG > 0 ) message("+ field loop ", j, " with name ", names(entity)[j])

          field   <- entity[[j]]
          value   <- field$value
          name    <- names(entity)[j]
          enabled <- field$enabled

          if ( enabled == FALSE )                              next    # skip disabled fields

          if ( ! is.character(value) )                         next    # only interested in sets of ⬈
strings
          if ( name == "builtin-remark" )                      next    # a mandatory field in most cases
          if ( name != "class" )                                       # class is always wanted
            {
              # split single strings
              split1 <- "[[:blank:]]+"                                 # define the split regex
              value  <- unlist(strsplit(value, split = split1))        # CAUTION: 'value' is overwritten

              # split if contains sub-entity
              #
              # assumes that "sub-id" associations are individual
              # or else the code (such as [1]) should be vectorized
              # (or confirmed to be thus)
              #
              split2 <- "\\."
              if ( length(value) == 1 ) {
```

```
              temp <- unlist(strsplit(value, split = split2))
              value        <- temp[1]
              names(value) <- temp[2]      # [1]
            }

            # set-theory test
            if ( length(intersect(value, entityIds)) == 0 ) next    # disjoint

            # integrity checks
            msg <- character()
            if ( ! is.element(entityId, entityIds)   ) msg[1] <- "coding error"
            if (   is.element(entityId, value)       ) msg[2] <- "self-referenced value"
            if (   length(setdiff(value, entityIds)) ) msg[3] <- "unresolved values"

            if ( length(msg) > 0 ) {                 # at least one of the above tripped
              msg <- msg[!is.na(msg)]                # remove NA element
              msg <- paste(msg, collapse = " / ")    # squish the rest into one string
              message("issues : ", msg)
              message("entity id '", entityId, "'")
              message("values:")
              noquote(value)
              message("entity ids:")
              noquote(entityIds)
              message("  warn : createBreadset : integrity check failure, see above")
              warning("integrity check failure (longer explanations earlier) ", msg)
            }
          }

          # load connection information
          bread[q <- q + 1] <- list(value)   # the value
          names(bread)[q]   <- name          # the field name
        }

      # skip
      if ( length(bread) < 1 ) {
        # indicates missing class data
        message("  warn : xem.createBreadset.1 : a length zero bread encountered", entityId)
        warning("integrity check failure (longer explanations earlier) ", "short bread")
        next
      }

      # skip the time horizon, it doesn't associate with anybody
      if ( entityId == "time-horizon" ) next

      # load entity
      if ( DBUG > 0 ) {
        message("about to insert a bread")
        str(bread)
      }

      breadset[[p <- p + 1]] <- bread        # insert
      names(breadset)[p]     <- entityId
    }

  # report
  if ( DBUG > 0 ) {
    message()
    message("i   : ", i)
    message("j   : ", j)
    message("p   : ", p)
    message("q   : ", q)
    message()
    message("breadset")
    str(breadset)
  }

  message("  info : xem.createBreadset.1 : complete, breadset length ", length(breadset))

  # return
  invisible(breadset)
}

# ---------------------------------
#  function : xem.createBreadset.2
# ---------------------------------
#  description  : takes conex1, converts to numerical indexes, and removes non-tail-and/or-head entitys
#  role         : breadplot data preparation
#  takes        : conex1
#  returns      : conex2
#  techniques   : 'match'
#  status       : complete
```

```
#
#  processed in xem file order - nested list sorting may require
#  dedicated code
#
# ---------------------------------

xem.createBreadset.2 <- function (vertexs)
{
  # report
  message("  info : xem.createBreadset.2 : commencing")

  # return early
  if ( length(vertexs) == 0 ) {              # got empty list
    message("  info : xem.createBreadset.1 : early return, vertexs length ", 0)
    return ( list() )                        # give empty list
  }
  if ( DBUG > 0 ) message()

  # obtain names
  vertexIds <- names(vertexs)

  # traverse and match
  for ( i in 1:length(vertexIds) )
    {
      vertexId <- vertexIds[i]

      if ( length(vertexs[[i]]) < 2 ) next   # only the class name present so skip

      for ( j in 2:length(vertexs[[i]]) )
        {
          value              <- vertexs[[i]][[j]]
          nums               <- match(value, vertexIds, nomatch = 0)  # non-default zero for no match
          vertexs[[i]][[j]]  <- nums
          names(vertexs[[i]][[j]]) <- names(value)     # CAUTION: required to carry over the "sub-id"
        }
    }

  # report
  if ( DBUG > 0 ) {
    message("vertexs")
    str(vertexs)
    message()
  }

  message("  info : xem.createBreadset.2 : complete, vertexs length ", length(vertexs))

  # return
  invisible(vertexs)
}

# ---------------------------------
#  function : xem.makeArrow
# ---------------------------------
#  description  : plots an "arrow", currently an "open head" segment
#  role         : called by 'xem.traverse'
#  takes        : see code
#  returns      : 'acolor' invisibly, which is a string like "#000000"
#  techniques   : 'rainbow', 'charmatch' and currently 'segments', 'points'
#  status       : complete - but coloration scheme probably needs work
#
#  last-match coloration scheme, based on class information
#
#  presumption               should usually be overwritten   --            gray
#  from "Cx"                 not currently implemented       link          blueish
#  to   "Cx"                 link                            link
#  to   "Cm"                 commodity                       link          .
#  from "Teas|Junc|Node"     to teas or gate                 connection    .
#  from "Asop"                                               link          .
#  to   "Asop"                                               link
#  from "Domain" to "Gate"   ie "ranked-sel-gates"           link
#  from "Overseer"                                           link          reddish
#
# ---------------------------------

xem.makeArrow <- function (conex,            # data structure
                           assoc,            # association index
                           tail,
                           head)
{
  # for my system (sojus), the following colorway worked well
  spectrum <- xem.colors
```

```
  # warn
  if ( head == 0 ) {
    message("  data : xem.makeArrow : null head found, tail-head-assoc ",
            tail, "-", head, "-", assoc)
  }
  if ( tail == 0 ) {
    message("  warn : xem.makeArrow : null tail found (very odd)")
    warning("null tail found (very odd)")
  }

  # grab class details
  outclass <- "(not specified)"              # tail entity class (later grouping)
  inclass  <- "(not specified)"              # head entity class (later grouping)
  if ( tail != 0 ) outclass <- conex[[tail]][["class"]]
  if ( head != 0 ) inclass  <- conex[[head]][["class"]]

  # process color
  tecs <- "^Teas|^Junc|^Node"
  if ( TRUE                                                ) acolor <- spectrum[ 1]
  if ( charmatch("Cx"      , outclass, nomatch = FALSE) ) acolor <- spectrum[10]
  if ( charmatch("Cx"      ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 9]
  if ( charmatch("Cm"      ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 8]
# if ( charmatch("Teas"    , outclass, nomatch = FALSE)
#      &&
#      charmatch("Teas"    ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 7]
  if ( length(grep(tecs, outclass))
       &&
       length(grep(tecs, inclass)) )                       acolor <- spectrum[ 7]
  if ((charmatch("Gate"    ,  inclass, nomatch = FALSE))
       ||
      (charmatch("Gate"    , outclass, nomatch = FALSE)
       &&
       length(grep(tecs, inclass)) ))                      acolor <- spectrum[ 6]
#      charmatch("Teas"    ,  inclass, nomatch = FALSE))) acolor <- spectrum[ 6]
  if ( charmatch("Asop"    , outclass, nomatch = FALSE) ) acolor <- spectrum[ 5]
  if ( charmatch("Asop"    ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 4]
  if ( charmatch("Domain"  , outclass, nomatch = FALSE)
       &&
       charmatch("Gate"    ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 3]
  if ( charmatch("Overseer", outclass, nomatch = FALSE) ) acolor <- spectrum[ 2]

  if ( head == 0 ) acolor <- spectrum[1]
  if ( tail == 0 ) acolor <- "black"         # quite noticeable

  # explain
  if ( DBUG > 0 ) message("  dbug : spanning ", outclass, " -> ", inclass, " with color ", acolor)

  # plot
  linewidth <- 1                             # line width, default = par("lwd"), probably 1
  linewidth <- 2                             # line width, default = par("lwd"), probably 1

  if ( head != 0 && tail != 0 )
    {
      segments(assoc, tail,                  # tail
               assoc, head,                  # head
               lwd = linewidth,
               col = acolor)
    }
  if ( tail != 0 )
    {
      points(assoc, tail,                    # tail
             pch = 21,                       # 21 = circle, 22 = square, 24 = up-triangle
             lwd = linewidth,
             bg  = acolor,
             col = acolor)

      # add hint about disabled head
      disabledTag <- "disabled"
      disabledTag <- "??"
      if ( head == 0 ) text(assoc, tail, labels = disabledTag, adj = c(0.5, -1.5))    # 'pos = 3' ⬋
overrides 'adj'
    }
  if ( head != 0 )
    {
      points(assoc, head,                    # head
             pch = 21,                       # 21 = circle, 22 = square, 24 = up-triangle
             lwd = linewidth,
             bg  = "white",
             col = acolor)
    }
```

```
  # process "sub-id" if present via the name attribute
  subid <- names(head)
  if ( ! is.null(subid) ) {                  # protect against no names
    if ( ! is.na(subid) ) {                  # protect against unassigned names
      text(assoc, head,
           subid,
           srt = 90,
           adj = c(0.5, -1.0),
           col = "lightslategray")
    }
  }

  # invisible return
  invisible(acolor)
}

# ---------------------------------
#  function : xem.breadTraverse
# ---------------------------------
#  description  : main conex traversal
#  role         : called by 'xem.breadboard'
#  takes        : 'conex'
#  returns      : arrow call count
#  techniques   : nested 'for', 'sort', 'rev'
#  status       : complete
# ---------------------------------

xem.breadTraverse <- function (conex)
{
  calls  <- 0                                # make call count (excludes empty association lists)
  acount <- 0                                # association count (includes empty association lists)
  for ( i in 1:length(conex) )
    {
      vertex <- conex[[i]]                   # grab out vertex

      if ( length(vertex) < 2 ) {            # data preparation problem
        message("  data : vertex lacks adjacency list (probably okay): ", names(conex)[i])
        next                                 # applies to innermost loop
      }

      for ( j in 2:length(vertex) )
        {
          adjlist <- vertex[[j]]             # graph theory list, numeric R vector
          acount  <- acount + 1
          if ( is.null(adjlist) ) {          # necessary protection against 'c()'
            message("  data : empty adjacency list (modeling issue): ", names(conex)[i], ".", ↙
names(vertex)[j])
            next                             # applies to innermost loop
          }

          adjlist <- xem.pyramidSort(adjlist, i)  # to keep the plot layering correct

          for ( k in 1:length(adjlist)  )
            {
              xem.makeArrow(conex,           # pass across data structure
                            acount,          # current association index
                            i,               # tail entity index
                            adjlist[k])      # head entity index

              calls <- calls + 1
            }
        }
    }

  # return
  invisible(calls)
}

# ---------------------------------
#  function : xem.breadboard
# ---------------------------------
#  description  : create a breadboard plot
#  role         : model visualization
#  takes        : data structure 'conex' and optional annotation and colorway arguments
#  returns      : void
#  techniques   : nested named list structure
#  status       : incomplete
#
#  in graph theoretic terms, 'conex' represents a
#  discriminated adjacency list with additional data --
```

```
#  comprising vertex labels, vertex class information,
#  and collective edge labels
#
#  in computer science terms, 'conex' is a two-deep
#  nested named list structure
#
#  the plot window layout is based on code from
#  plot function 'dotchart'
#
# ---------------------------------

xem.breadboard <- function (conex,                # discriminated adjacency list with additional data
                            main = "anonymous",   # optional main label (arrow count later added ⬋
underneath)
                            cex  = par("cex"))    # character magnification (NOT passed to ⬋
'xem.makeArrow')
{
  # reporting
  message("  info : xem.breadboard : commencing")

  # some integrity checks
  if ( !is.list(conex) ) stop("conex is not a list")
  if ( length(conex) == 0 ) {
    message("  warn : xem.breadboard : conex is empty")
    warning("conex is empty")
  }

  # modify graphical parameters
  opar <- par("mai",                              # margin sizes in inches for c(bottom, left, top, right)
              "xaxs",                             # x-axis intervals
              "yaxs")                             # y-axis intervals
  on.exit(par(opar))
  par(xaxs     = "r")                             # "i" is without the 4% extension utilized by "r"
  par(yaxs     = "r")                             # "i" is without the 4% extension utilized by "r"

  # open a plot
  plot.new()                                      # create or start a new plot frame

  # define some fallbacks
  arrowcalls <- 0
  ylabels <- character()
  xlabels <- character()

  if ( length(conex) > 0 ) {                      # protect against non-data

    # extract label data
    entitys <- names(conex)
    assocs  <- character()                        # length zero string vector
    for ( i in 1:length(conex) )
      {
        assocs <- c(assocs, names(conex[[i]])[-1])     # skip class (the "[-1]")
      }

    # add leading counts
    if ( length(entitys) > 0 ) entitys <- sprintf("%02d \xb7 %s", seq(length(entitys)), entitys)
    if ( length(assocs)  > 0 ) assocs  <- sprintf("%02d \xb7 %s", seq(length(assocs)) , assocs)

    # more processing with empty list protection
    ylabels <- entitys
    xlabels <- assocs
    if ( length(ylabels) == 0 ) ylabels <- ""
    if ( length(xlabels) == 0 ) xlabels <- ""
    y       <- length(ylabels)
    x       <- length(xlabels)

    # reset margins
    linch   <- max(strwidth(ylabels,              # left margin in inches
                            unit = "inch"))
    nmai    <- par("mai")
    nmai[2] <- nmai[4] + linch + 0.0              # was + 0.1
    par(mai = nmai)

    binch   <- max(strwidth(xlabels,              # bottom margin in inches
                            unit = "inch"))
    nmai    <- par("mai")
    nmai[1] <- nmai[4] + binch + 0.0
    par(mai = nmai)

    # set and add axes
    ylim <- c(1, y)
    xlim <- c(1, x)
```

```
    plot.window(xlim = xlim, ylim = rev(ylim), log = "")

    height  <- par("csi")                    # character height in inches
    color   <- par("fg")

    loffset <- (linch + 0.1)/height
    mtext(ylabels,
          side = 2,
          line = loffset,
          at   = 1:y,
          adj  = 0,
          col  = color,
          las  = 2,
          cex  = cex)

    boffset <- (binch + 0.1)/height
    mtext(xlabels,
          side = 1,
          line = boffset,
          at   = 1:x,
          adj  = 0,
          col  = color,
          las  = 2,
          cex  = cex)

    abline(h   = 1:y,                        # horizontal gridlines
           lty = "dotted",
           col = "lightgray")

    # traverse data structure and add arrows (the real work)
    arrowcalls <- xem.breadTraverse(conex)   # traversal call

    # debugging support
    if ( DBUG > 1 ) {
      message("  info : adding axes and box to plot")
      axis(3)                                # add an axis to a plot, 3 = top
      axis(4)                                # add an axis to a plot, 4 = right
      box()                                  # draw a box around a plot
    }

  } # empty data protection

  # finalize and print main title
  main <- paste(main)
  title(main = main, xlab = "", ylab = "")

  # finalize and print ancillary message
  msg    <- character()
  msg[1] <- paste("xeona best mode =", xmode)
  msg[2] <- paste("associations =", as.character(arrowcalls))
  if ( xmode == "na" ) msgs <- msg[2]
  else                 msgs <- paste(msg, collapse = " / ")
  mtext(msgs, col = "darkslategray")

  message("  info : xem.breadboard : display ", msgs)
  message("  info : xem.breadboard : complete, arrow count ", arrowcalls)

  # housekeeping
  invisible(list(entitys      = ylabels,
                 associations = xlabels,
                 arrowcount   = arrowcalls))

}

# ---------------------------------
#  function : xem.breadplot
# ---------------------------------
#  description  : wrapper to the core calls
#  role         : job 8 point of contact
#  techniques   : see core calls
#  status       : complete
#
#  'cex' defaults to 1, useful values can be 0.9 and 0.8
#
# ---------------------------------

xem.breadplot <- function (recset,                # recset
                           main = "anonymous",    # optional main label (arrow count later added ⬀
underneath)
                           cex  = par("cex"))     # character magnification (NOT passed to ⬀
'xem.makeArrow')
```

```
{
  # reporting
  message("  info : xem.breadplot : commencing")
  on.exit(message("  info : xem.breadplot : complete (on.exit)"))

  # code
  conex1  <- xem.createBreadset.1(recset)
  conex2  <- xem.createBreadset.2(conex1)
  capture <- xem.breadboard(conex2, main = main, cex = cex)

  # return
  invisible(capture)
}

# ---------------------------------
#  function : xem.breadplot.info
# ---------------------------------
#  description  : explain the prevailing color scheme
#  role         : strictly informational
#  status       : ongoing
# ---------------------------------

xem.breadplot.info <<- function ()           # CAUTION: exported
{
  msg <- "
current status (currently outdated)

 1   gray              starting point

 2   purple            Cx       out
 3   dark-blue         Cx        in
 4   mid-blue          Cm        in
 5   blue-green        Teas     out                ambient-air-context, socket-electricity-commodity
 6   dark-blue         Gate     connection
 7   leaf-green        Asop     out                technical-assets
 8   yellow-green      Asop      in                asset-operators
 9   orange            Domain   out && Gate in     ranked-sel-gates
10   red               Overseer out                ranked-orig-domains

 1   gray              head absent
 0   black             tail absent (error)

"
  cat(msg)
  invisible(msg)
}

# ---------------------------------
#  function : tst.breadboard
# ---------------------------------

tst.breadboard <- function (dbug)
{
  DBUG <<- dbug
  message("  * * * * * * * * * *\n  test : breadboard test commencing / DBUG = ", DBUG)

  conex <<-
    list("overseer"             = list(class                = "Overseer"             ,
                                       "ranked-orig-domains"  = c(2   )                ) ,
         "domain-controller-1"  = list(class                = "DomainController"     ,
                                       "ranked-sel-gates"     = c(    )                ,   # note 'c()'
                                       "asset-operators"      = c(3, 4)                ) , # 4 is for ⬋
test purposes
         "asop-null-1"          = list(class                = "AsopNull"             ,
                                       "technical-assets"     = c(    )                ) ,
         "cx-ambient-air-sim-1" = list(class                = "CxAmbientAirSim"      ) )

  ids <- names(conex)
  cxs <- character()
  for ( i in 1:length(conex) )
    {
      cxs <- c(cxs, names(conex[[i]])[-1])   # skip class (the "[-1]")
    }

  print(ids)
  print(cxs)

  capture <<- xem.breadboard(conex,
                             main = paste("associations test plot / DBUG = ", DBUG),
                             cex  = 1.0)      # optional, scales 'mtext' labels only (not passed to ⬋
'xem.makeArrow')
```

```
  print(capture$entitys)
  print(capture$associations)
  print(capture$arrowcount)

  str(conex)                                  # CAUTION: neither 'cat' nor 'print' (which adds final ⬈
NULL) needed

  message("  test : breadboard test complete")
}

# ---------------------------------
#  junk
# ---------------------------------
#
#  print(lapply(conex, names))
#
#     > Lst[5] <- list(Mat)
#     > names(Lst)[5] <- "matrix"
#
#  this adds the entire field: bread[[n <- n+1]] <- entity[["class"]]

#  $Id: breadplot.R 5089 2010-08-16 17:51:55Z robbie $
#  end of file
```

```
#  file-purpose     : build nested dataset from xem file data
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Thu 04-Jun-2009 19:55 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 2925 $
#  $Date: 2009-06-22 18:03:07 +0200 (Mon, 22 Jun 2009) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/dataproc.R $

#  OVERVIEW
#
#  This code uses a nested list structure with named
#  components.  In pseudo-UML:
#
#                *             *            1
#    recset <>---- record <>---- field <>---- value
#    |             |             |
#    + modelend    + enabled     + enabled
#                  + kind        + kind
#
#      where * indicates a multiplicity of 0..*
#
#  Non-mandatory components, which arise from the
#  optional xem field remark duly split (see code for
#  details), are omitted from the above diagram.
#
#  Metadata, which is added here, is also omitted from
#  the above diagram.
#
#  Individual sub-objects can be dereferenced as
#  follows, with '$' being the R "list subset"
#  operator:
#
#    x <- varname  $ identifier  $ fieldname  $ ..
#
#  Specifically, the 'recset' is a workspace variable
#  referred to by a symbol in the normal way.
#
#  The 'record' component is named using its
#  'identifier' with the dot-separated prefix
#  ("program."  or "entity.") duly removed.  The record
#  'kind' can be {"program", "entity"} and thus retains
#  this information.
#
#  The field 'kind' can be {"in", "out"}.
#
#  The record and field 'enabled' are {TRUE, FALSE}, as
#  are the recset 'modelend'
#
#  The 'value' may be -- in terms of R -- of type
#  logical, string, or double (with classic ints
#  treated as double) and may be a single element
#  vector (a scalar) or a multi-element vector.
#
#  ACCESS
#
#  Access a recset/record/field/value as follows (and
#  note single or double quotes or back-ticks required
#  for names containing '-' chars):
#
#    > rrfv <- rs$"entity-1"$"field-ts"$value
#    > rrfv <- getValue(rs, "entity-1", "field-ts")
#
#  CODING ISSUES
#
#  Regarding the recursive indexing of nested lists,
#  see:
#
#    > ?Extract
#
#  In particular, '[' omits the component name (if
#  present), whereas '[[' does not.  Moreover, '[[' and
#  '$' can only select a single element, whereas '['
#  can select several (by submitting an integer
#  vector).  In addition, both '[' and '[[' accept
#  computed indexes (via symbols), whereas '$' does
#  not.  In most environments, '[[' and '$' allow
#  partial (first encountered) matching.
```

```
#
#  Component names are forced onto list components
#  using the 'names' call, see:
#
#    > ?names

# ---------------------------------
#  debug settings
# ---------------------------------

# 'WARNSET' is used in 'as.numeric' lexical cast calls
# negative is suppress warnings, zero is store warnings

WARNSET <- -1

# ---------------------------------
#  function : xem.delist
# ---------------------------------
#  description  : unlist a list or timeseries raw string
#  role         : data processing
#  takes        : quote-separated list or timeseries as raw string
#  returns      : string vector
#  techniques   : 'strsplit'
#  status       : complete
#
#  the 'xeona' "pairwise delimiter for string values"
#  is assumed to be defined as a double quote
#
#  yields NA if 'rawString' is empty, that is, no
#  pairwise delimiters are present -- hence the minimum
#  sensible input is the string "\"\""
#
#  examples
#
#    c("")                           gives   NA
#    c("\"\"")                       gives   c("")
#    c("\"one two\"")                gives   c("one two")
#    c("\"one two\" \"eins zwei\"")  gives   c("one two", "eins zwei")
#
# ---------------------------------

xem.delist <- function (rawString)
{
  # define the split regex
  sep <- "\"[[:blank:]]*\""                 # CAUTION: note the * (and not +) repetition operator

  # integrity check
  if ( ! is.character(rawString) ) {
    message("  warn : xem.delist : 'rawString' argument is not of type character (structure follows)")
    warning("'rawString' argument is not of type character")
    print(str(rawString))
  }

  # main code
  buff      <- unlist(strsplit(rawString, split = sep))     # CAUTION: 'unlist' is essential
  len       <- length(buff)
  buff[1]   <- substr(buff[1]  , 2, nchar(buff[1])      )   # trim leading quote
  buff[len] <- substr(buff[len], 1, nchar(buff[len]) - 1)   # trim trailing quote

  # return
  buff
}

# ---------------------------------
#  function : xem.unitsThintRemark
# ---------------------------------
#  description  : splits a " [-] x remaining remark" string
#  role         : recset building
#  takes        : single string
#  returns      : string vector (unnamed)
#  techniques   : 'grep'
#  status       : complete
#
#  expected format (optional) : [-] x remaining remark
#  interpretation             : units / type-hint / remaining-remark
#
#  multiple spaces in the remaining remark are
#  collapsed into one (think of this as a feature)
#
# ---------------------------------
```

```
xem.unitsThintRemark <- function (remain)
{
  output <- c(NA,                           # optional units
              NA,                           # optional type hint
              NA)                           # optional remaining remark

  sep    <- "[[:blank:]]+"                  # one or more space and/or tab characters
  buffer <- unlist(strsplit(remain, sep))

  # process optional units field
  if ( length(grep("^\\[.+\\]$", buffer[1])) )
    {
      # pop buffer
      output[1] <- buffer[1]
      buffer    <- buffer[-1]
    }

  # process optional type hint field
  if ( length(grep("^[biflsxBIFLSX]$", buffer[1])) )
    {
      # pop buffer
      output[2] <- buffer[1]
      buffer    <- buffer[-1]
    }

  # process optional remaining remark
  if ( length(buffer) )
    {
      output[3] <- paste(buffer, sep = " ", collapse = " ") # CAUTION: string not vector required
    }

  # return
  output
}

# ---------------------------------
#  function : xem.loop
# ---------------------------------
#  description  : contains main processing loop
#  role         : build call
#  takes        : line-oriented xem model 'data'
#  returns      : equivalent 'recset' data structure
#  techniques   : nested named lists (multi-type vectors)
#  status       : complete
#
#  acceptable construct: "if ( length(grep(..)) )"
#  alternative: "if ( any(i <- grep(..)) )" # the 'i' vector can then be used
#
# ---------------------------------

xem.loop <- function (data,
                      xemscope = 1:length(data))
{
  # reporting
  message("  info : xem.loop : commencing")
  on.exit(  message("  info : xem.loop : complete (on.exit)"))
  if ( DBUG > 0 ) message()

  # nested datatype
  recset <- list(modelend = FALSE)

  # separators
  sepA <- "\\."                             # dot char in regex format
  sepB <- "[<>]"                            # either a '<' or a '>' character
  sepC <- "[[:blank:]]+"                    # one or more space and/or tab characters

  # associated count variables
  rcnt  <- length(recset)                   # record counter
  fcnt  <- 0                                # field counter (should be reset)

  # variables for completion reporting
  records     <- 0
  entitys     <- 0
  disEntitys  <- 0
  fields      <- 0
  disFields   <- 0
  blanks      <- 0
  comments    <- 0
  loops       <- 0

  # meta data
```

```
ids  <- NULL

# loop
for ( i in xemscope )
  {
    loops  <- loops + 1
    line   <- data[i]
    line   <- xem.trim(line)
    if ( DBUG >  2 ) if ( nchar(line) > 0 ) message("LINE ", line)

    # OPTION: a blank line
    if ( 0 == nchar(line) )
      {
      blanks  <- blanks + 1
    }

    # OPTION: the end of the model
    else if ( length(grep("^model-end", line)) )
      {
        recset[["modelend"]] <- TRUE
        break
      }

    # OPTION: a record
    else if ( length(grep("^(#[[:blank:]]*|)(program|entity)\\.", line)) )
      {
        # identify and remove any column zero disable char
        if ( charmatch("#", line, nomatch = FALSE) ) {
          line <- substr(line, 2 , nchar(line))      # strip disable char
          line <- xem.trim(line)                     # trim any leading spaces
          enabled <- FALSE
          records <- records + 1
        }
        else {
          enabled <- TRUE
          records <- records + 1
        }

        # process core information
        two      <- unlist(strsplit(line, sepA))     # 'unlist' transforms "list" to "character"
        entid    <- two[2]

        if      ( length(grep("program\\.", line)) ) kind <- "program"
        else if ( length(grep("entity\\." , line)) ) kind <- "entity"
        else warning("coding error 01")

        if ( kind == "entity" ) {
          if ( enabled ) entitys    <- entitys    + 1
          else           disEntitys <- disEntitys + 1
          ids[length(ids) + 1]      <- entid         # meta information
        }

        # load
        record   <- list(enabled = enabled,
                         kind    = kind)

        rcnt     <- rcnt + 1
        fcnt     <- length(record)                  # reset the field counter appropriately

        recset[[rcnt]]      <- record               # insert record stub
        names(recset)[rcnt] <- entid                # associate name

        if ( DBUG > 0 ) message("+ record push: ", names(recset)[rcnt])
      }

    # OPTION: a field
    else if ( length(grep(sepB, line)) )
      {
        # identify and remove any leading disable char
        line <- xem.trim(line)
        if ( charmatch("#", line, nomatch = FALSE) ) {
          line <- substr(line, 2 , nchar(line))       # strip disable char
          enabled   <- FALSE
          disFields <- disFields + 1
        }
        else {
          enabled   <- TRUE
          fields    <- fields + 1
        }

        # process core information
```

```
          two      <- unlist(strsplit(line, sepB))     # 'unlist' transforms "list" to "character"
          left     <- xem.trim(two[1])                 # recover left hand part and trim
          lefts    <- unlist(strsplit(left, sepC))     # split
          fname    <- lefts[1]                         # grab first element
          fremain  <- lefts[-1]                        # grab the rest

          if      ( length(grep(">", line)) ) kind <- "in"
          else if ( length(grep("<", line)) ) kind <- "out"
          else warning("coding error 02")

          output   <- xem.unitsThintRemark(fremain)    # local function
          funits   <- output[1]
          fthint   <- output[2]
          fremark  <- output[3]

          right    <- xem.trim(two[2])                 # recover right hand part and trim
          buffer1  <- unlist(strsplit(right, sepC))    # split

          prior    <- options(warn = WARNSET)          # negative is suppress warnings, zero is store ⬋
warnings
          buffer2  <- as.numeric(buffer1)              # string to number coercion
          options(warn = prior$warn)                   # restore warnings

          # process right data
          if ( fname == "class" ) {                    # class value is never given in quotes
            fvalues <- right                           # revert
          }
          else if ( any(is.na(buffer2)) ) {            # the lexical cast failed
            fvalues <- xem.delist(right)               # split string timeseries but NOT entity lists
          }
          else {
            fvalues <- buffer2                         # accept the lexical cast
          }

          if ( DBUG >  1 ) {
            message("fremain (one)  : ", toString(fremain))
            message("fvalues        : ", fvalues)
          }

          # load
          fcnt      <- fcnt + 1
          record    <- recset[[rcnt]]
          field     <- list(enabled = enabled,
                            kind    = kind,
                            units   = funits,
                            thint   = fthint,
                            remark  = fremark,
                            value   = fvalues)

          record[[fcnt]]      <- field
          names(record)[fcnt] <- fname
          recset[[rcnt]]      <- record

          if ( DBUG > 1 ) message("  - info       (", rcnt, "/", fcnt,"): ", paste(fname, fremain, ⬋
right))
          if ( DBUG > 0 ) message("  + field push (", rcnt, "/", fcnt,"): ", names(record)[fcnt])
        }

      # OPTION: a comment
      else                                             #  by process of elimination
        {
          comments <- comments + 1
        }
    }

  # meta data lead-up
  timestamp <- format(Sys.time(), "%d-%b-%Y %H:%M:%S")

  # add meta data
  rcnt                <- rcnt + 1
  recset[[rcnt]]      <- list(timestamp = timestamp,
                              entitys   = entitys,
                              ids       = ids)
  names(recset)[rcnt] <- "metadata"                    # associate name

  if ( DBUG > 0 ) message("+ metadata push: ", names(recset)[rcnt])
  if ( DBUG > 0 ) message()

  # completion reporting

  message("entities       : ", sprintf("%4d / %d", entitys, disEntitys))
```

```
  message("fields         : ", sprintf("%4d / %d", fields , disFields))
  message("comments       : ", sprintf("%4d", comments))
  message("blanks         : ", sprintf("%4d", blanks))
  message("loops          : ", sprintf("%4d", loops))

  if ( recset$"modelend" == TRUE  ) message("model end      : found")
  else                              message("model end      : not encountered")
  if ( recset$"modelend" == FALSE ) warning("no model end marker found")

  # final return
  invisible(recset)
}

# ---------------------------------
#  function : tst.delist
# ---------------------------------

tst.delist <- function ()
{
  inputs <- c("",
              "\"\"",
              "\"one two\"",
              "\"one two\" \"eins zwei\" \"alpha beta\"")
  for ( i in 1:length(inputs) )
    {
      input  <- inputs[i]
      output <- xem.delist(input)
      cat(sprintf("  test : %-40s  %-40s  %3d\n", input, output[1], length(output)))
    }
}

# ---------------------------------
#  function : tst.dataproc
# ---------------------------------

tst.dataproc <- function (xemfile,
                          dbug)
{
  DBUG <<- dbug
  message("  * * * * * * * * * *\n  test : data processing test (largely hollow) commencing / DBUG = ↙
", DBUG)

  tst.delist()

  message("  test : data processing test complete")
}

# ---------------------------------
#  junk
# ---------------------------------

#   # CAUTION: line ranges refer to the processed XEM file
#   xemscope <- 50:60                       # just entity of class 'Overseer'
#   xemscope <- 76:99                       # just entity of class 'CxAmbientAirSim'

#  $Id: dataproc.R 2925 2009-06-22 16:03:07Z robbie $
#  end of file
```

```
#  file-purpose     : extract data and plot timeseries
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Fri 05-Jun-2009 08:01 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 5089 $
#  $Date: 2010-08-16 19:51:55 +0200 (Mon, 16 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/extract.R $

#  SUPPLIES
#
#      xem.summarize
#
#      xem.getPolicy
#      xem.getPlotList
#
#      xem.getSteps
#      xem.getInterval
#      xem.getClass
#      xem.getKind
#      xem.getUnits
#      xem.getValue
#
#      xem.getCounts
#      xem.examineEntity     (incomplete)
#      xem.plotValue
#      xem.plotAllTs
#
#      tst.extract

# ---------------------------------
#  function : xem.summarize
# ---------------------------------
#  description  : summarizes dataset using a plot frame
#  role         : job 1 point of contact
#  takes        : 'recset'
#  returns      : data (with any formatting blanks removed)
#  techniques   : 'text'
#  status       : complete
# ---------------------------------

xem.summarize <- function (recset)
{
  # reporting
  message("  info : xem.summarize : commencing")
  on.exit(message("  info : xem.summarize : complete (on.exit)"))

  # overall plot
  plot.new()
  title(xtitle)
  mtext("model summary")

  # dedicated 'load' function
  i <- 0                                    # declared to prevent 'i' from leaking further
  load <- function(tag = "", value = "")
    {
      i         <<- length(tags) + 1
      tags[i]   <<- tag
      values[i] <<- as.character(value)
    }

  # specify the summary data
  tags   <- character()
  values <- character()

  if ( is.null(xeona) ) xeona <- "na"        # correction for when 'xeona' is NULL

  load("xeona"                , xeona)
  load("xemfile"              , xemfile)
  load("xeona call"           , xcall)
  load()
  load("jobs"                 , paste(jobs, " = ", jobMsgs))
  load("xeona best mode"      , xmode)
  load("model end"            , recset$"modelend")     # logical
  load("svn"                  , xver)
  load()
  load("steps"                , xem.getSteps(recset))
  load("interval"             , xem.getInterval(recset))
```

```
  load()
  load("enabled entities"     , xem.getCounts(recset)["enabled-entitys"])
  load("disabled entities"    , xem.getCounts(recset)["disabled-entitys"])
# load("overall data"         , xem.getCounts(recset)["value-fields"])
  load("timeseries data"      , xem.getCounts(recset)["timeseries-fields"])
  load("singles data"         , xem.getCounts(recset)["single-fields"])

  # adjusment and plotting
  hinch <- par("csi")                       # character height in inches
  winch <- max(strwidth(tags))
  v0 <- (0.22 * hinch * 1:i) + 0.3          # final is bottom up adjust
  h1 <-  0.0                                # value is rightwards adjust
  h2 <-  h1 + 0.06 + winch
  text(h1, v0, rev(tags)  , adj = c(0, 0), xpd = TRUE, font = 2)
  text(h2, v0, rev(values), adj = c(0, 0), xpd = TRUE)

  # export data
  namedvec        <- values
  names(namedvec) <- tags
  namedvec        <- namedvec[names(namedvec) != ""]   # skip the blank line calls on export

  # return
  invisible(namedvec)
}

# ---------------------------------
#  function : xem.splitFieldName
# ---------------------------------

xem.splitFieldname <- function (fieldname,
                                prefix = "entity")
{
  parts <- unlist(strsplit(fieldname, split = ".", fixed = TRUE))
  if ( length(parts) == 0 ) {
    message("  warn : xem.splitFieldname : empty field name")
    warning("empty field name")
    parts <- NA
  }
  else if ( parts[1] != prefix ) {
    message("  warn : xem.splitFieldname : field name without required qualifier")
    warning("field name without required qualifier '", fieldname, "'")
    parts <- NA
  }
  else if ( length(parts) != 3 ) {
    message("  warn : xem.splitFieldname : malformed field name")
    warning("malformed field name '", fieldname, "'")
    parts <- NA
  }
  return(parts)
}

# ---------------------------------
#  function : xem.getPolicy
# ---------------------------------

xem.getPolicy <- function (recset)
{
  policy <- recset$"r-processing"$"r-policy"$value
  as.integer(policy)                        # already cast to numeric and known to be an integer
}

# ---------------------------------
#  function : xem.getPlotList
# ---------------------------------

xem.getPlotList <- function (recset)
{
  recset$"r-processing"$"r-plot-list"$value
}

# ---------------------------------
#  function : xem.getSteps
# ---------------------------------

xem.getSteps <- function (recset)
{
  xem.getValue(recset, "time-horizon", "steps")
}

# ---------------------------------
#  function : xem.getInterval
```

```
# ---------------------------------

xem.getInterval <- function (recset)
{
  xem.getValue(recset, "time-horizon", "interval")
}

# ---------------------------------
#  function : xem.getClass
# ---------------------------------

xem.getClass <- function (recset,
                          identifier)
{
  xem.getValue(recset, identifier, "class")
}

# ---------------------------------
#  function : xem.getKind
# ---------------------------------

xem.getKind <- function (recset,
                         identifier,
                         fieldname)
{
  recset[[identifier]][[fieldname]]$kind
}

# ---------------------------------
#  function : xem.getUnits
# ---------------------------------

xem.getUnits <- function (recset,
                          identifier,
                          fieldname)
{
  recset[[identifier]][[fieldname]]$units
}

# ---------------------------------
#  function : xem.getValue
# ---------------------------------

xem.getValue <- function (recset,
                          identifier,
                          fieldname)
{
  recset[[identifier]][[fieldname]]$value
}

# ---------------------------------
#  function : xem.getCounts
# ---------------------------------

xem.getCounts <- function (recset)
{
  # grab names directly
  entitys <- recset$"metadata"$"ids"

  # define counters
  yesEntitys    <- 0                        # not counting 'entity.time-horizon'
  noEntitys     <- 0
  cntValues     <- 0
  cntTimeseries <- 0
  cntSingles    <- 0
  cntNA         <- 0                        # should normally remain zero

  # traverse dataset
  for ( i in 1:length(entitys) )
    {
      id     <- entitys[i]
      entity <- recset[[id]]                # grab entity
      if ( id == "time-horizon" ) next
      if ( entity[["enabled"]] == TRUE ) yesEntitys = yesEntitys + 1
      else                                noEntitys =  noEntitys + 1

      for ( j in 1:length(entity) )
        {
          field <- entity[[j]]              # grab field
          if ( length(field) != 6 ) next
          value <- field[["value"]]
```

```
          cntValues                                    <- cntValues     + 1
          if      ( any(is.na(value))  ) cntNA         <- cntNA         + 1
          else if ( length(value)  > 1 ) cntTimeseries <- cntTimeseries + 1
          else if ( length(value) == 1 ) cntSingles    <- cntSingles    + 1
          else {
            warning("strange value ", value)  # defensive coding
          }
        }
    }

  # load data
  data <- numeric()
  data <- c(data, "enabled-entitys"   = yesEntitys)
  data <- c(data, "disabled-entitys"  =  noEntitys)
  data <- c(data, "value-fields"      = cntValues)
  data <- c(data, "timeseries-fields" = cntTimeseries)
  data <- c(data, "single-fields"     = cntSingles)

  # return
  invisible(data)
}

# ---------------------------------
#  function : xem.examineEntity
# ---------------------------------

xem.examineEntity <- function (recset,
                               identifier)
{
  x <- recset[[identifier]]
  cat("str(ucture):\n")
  str(x)                                    # compactly display structure of an object
  cat("summary:\n")
  print(summary(x))                         # summarize an object

  message()
  class <- xem.getClass(recset, identifier)
  message("class           : ", class)
  builtinRemark <- xem.getValue(recset, identifier, "builtin-remark")
  message("builtin remark  : ", builtinRemark)
  fieldCount <- length(recset[[identifier]])
  message("distinct fields : ", fieldCount - 1)
  fieldNames <- names(recset[[identifier]])
  if ( fieldCount > 2 )
    message("distinct names  : ", fieldNames[2])
  for ( i in 3:fieldCount )
    message("                  ", fieldNames[i])
  message()

  # TOFIX: examineEntity: extend field reporting (not important at present)
}

# ---------------------------------
#  function : xem.plotValue
# ---------------------------------

xem.plotValue <- function (recset,
                           identifier,
                           fieldname)
{
  # obtain timeseries
  y <- xem.getValue(recset, identifier, fieldname)

  # integrity check
  if ( is.null(y) ) {
    msg <- paste("cannot plot timeseries (check name)        ", identifier, ".", fieldname)
    plot.new()
    text(0.5, 0.5, labels = msg, cex = 1.3, col = "orange")
    return(numeric())                       # CAUTION: do not use 'invisible' for early returns
  }

  # zero-base the step count (because that is how 'xeona' works)
  steps <- xem.getSteps(recset)
  x     <- 0:(steps - 1)                    # CAUTION: incorrect code: xlim = c(0, steps-1)

  # some defensive coding
  if ( steps < 2 ) {
    message("  warn : xem.plotValue : timeseries of length ", steps)
    warning("timeseries of length ", steps)
  }
```

```
  # label processing
  class     <- xem.getClass(recset, identifier)
  inv       <- xem.getValue(recset, "time-horizon", "interval")
  kind      <- xem.getKind(recset, identifier, fieldname)

  mean      <- signif(mean(y), 4)
  units     <- xem.getUnits(recset, identifier, fieldname)
  fname     <- sub("ies$", "y", fieldname)        # remove trailing pluralization if present
  fname     <- sub("s$"  , "" , fname)            # remove trailing pluralization if present

  classtag  <- paste("class ="       , class)
  invaltag  <- paste("interval ="    , inv, "s")   # simply assume units of 's'
  kindtag   <- paste("kind ="        , kind)
  meantag   <- paste("mean ="        , mean)
  modetag   <- paste("xeona best mode =", xmode)
  steptag   <- paste("steps ="       , steps)
  vertag    <- paste("svn version =" , xver)
  xtag      <- paste("step count")

  if ( is.na(units) ) ytag <- fname
  else                ytag <- paste(fname, units)

  mlabel    <- substitute(paste( identifier, " ", phi, " ", fieldname))    # recovers parts!
  mlabel    <- expression(paste( identifier, " ", phi, " ", fieldname))
  mlabel    <- paste(identifier, fieldname, sep = " \xb7 ")                # ASCII 183 middle dot

  if ( xver == 0 ) tlabel <- paste(classtag, kindtag, modetag, sep = " / ")
  else             tlabel <- paste(classtag, kindtag, modetag, vertag ,sep = " / ")
  xlabel    <- paste(xtag, invaltag, steptag, sep = " / ")
  ylabel    <- paste(ytag, sep = " / ")
  slabel    <- paste(vertag, sep =" / ")

  # plot call
  # 'bty' box type usefully in { "o", "l", "n" }
  # 'ylim' controls y-axis limits
  plot.new()
  # plot(x, y, xlab = xlabel, ylab = ylabel, las = 1, bty = "l")                    # y-axis starts ⬋
min(y)
  plot(x, y, ylim = c(0, max(y)), xlab = xlabel, ylab = ylabel, las = 1, bty = "l")  # y-axis starts ⬋
zero
  title(main = mlabel)
  mtext(text = tlabel)
  abline(h = mean(y), lty = "dotted", col = "lightgray")
  # alternative label: text(x = 0, y = mean(y), adj = c(0, -1), labels = "mean")
  mtext(side = 4, at = mean(y), las = 0, text = meantag)

  # return invisibly
  invisible(y)
}

# ---------------------------------
#  function : xem.multiplot
# ---------------------------------
#
#  currently uses 'par(mfrow)', but could equally use
#  'layout' or 'trellis' if more sophistication is
#  required
#
# ---------------------------------

xem.multiplot <- function (recset,
                           flist)            # field list to plot
{
  # reporting
  message("  info : xem.multiplot : commencing, plots, ", length(flist))
  on.exit(message("  info : xem.multiplot : complete (on.exit), plots ", subs))

  # number of subplots
  subs <- length(flist)

   # rc "algorithm"
  if      ( subs <=  1 ) plots <- c(1,2)
  else if ( subs <=  2 ) plots <- c(1,2)
  else if ( subs <=  4 ) plots <- c(2,2)
  else if ( subs <=  6 ) plots <- c(2,3)
  else if ( subs <=  9 ) plots <- c(3,3)
  else if ( subs <= 12 ) plots <- c(3,4)
  else if ( subs <= 16 ) plots <- c(4,4)
  else if ( subs <= 20 ) plots <- c(5,4)
  else {
    message("  warn : xem.plotAllSomeTs : too many plots, plot flist truncated, was ", subs)
```

```
    warning("plot flist truncated")
    flist <- flist[1:20]
    plots <- c(5,4)
  }

  cntPlots <- plots[1] * plots[2]
  if     ( cntPlots <= 4 ) mycex <- 1.0
  else                     mycex <- 0.7

  # prepare plot
  opar <- par(no.readonly = TRUE)
  plot.new()
  par(mfrow = plots)

  # recover data and plot
  if ( subs > 0 ) {                         # protect against empty feild lists
    for ( k in 1:length(flist) )
      {
        out    <- flist[k]                  # use as label also
        part   <- unlist(strsplit(out, split = ".", fixed = TRUE))
        ts     <- xem.getValue(recset, part[2], part[3])
        mlabel <- paste(part[2], part[3], sep = " \xb7 ")   # ASCII 183 middle dot

        plot(ts,                            # timeseries as unnamed numeric vector
             main = mlabel,                 # see above
             xlab = "", ylab = "",          # suppress axis labels
             pch  = 3,                      # 3 = a cross
             cex  = mycex)                  # reduce plot character as determined above
        mtext(text = paste("plot", k),
              padj = -0.5,                  # 'padj' is perpendicular, negative is raise
              cex  = mycex)
      }
  }

  # housekeeping
  par(opar)

  # return
  invisible(subs)
}

# ---------------------------------
#  function : xem.plotAllTs
# ---------------------------------

xem.plotAllTs <- function (recset,
                           kind)            # 'kind' in { "in", "out" }
{
  # reporting
  message("  info : xem.plotAllTs : commencing")
  on.exit(message("  info : xem.plotAllTs : complete (on.exit), kind ", kind))

  # grab names directly and traverse dataset
  entitys   <- recset$"metadata"$"ids"
  xins     <<- character()
  xouts    <<- character()
  for ( i in 1:length(entitys) )
    {
      id     <- entitys[i]
      entity <- recset[[id]]                # grab entity
      for ( j in 1:length(entity) )
        {
          fname <- names(entity[j])
          field <- entity[[fname]]          # grab field
          if ( length(field) != 6 )   next  # not a data field
          value <- field[["value"]]
          kind2 <- field[["kind"]]
          if ( ! is.numeric(value) )  next  # not numeric data
          if ( length(value)  < 2 )   next  # not a timeseries
          if ( kind2 == "in"  ) xins  <<- c(xins,  paste("entity", id, fname, sep = "."))
          if ( kind2 == "out" ) xouts <<- c(xouts, paste("entity", id, fname, sep = "."))
        }
    }

  # remove (as it happens adjacent) duplicate entries
  # from 'xins' and 'xouts' -- these arise from
  # additional disabled fields on the XEM file
  xins  <<- unique(xins)
  xouts <<- unique(xouts)

  # switch on kind and then plot
```

```
  switch (kind,
          "in"  = xem.multiplot(recset, xins),
          "out" = xem.multiplot(recset, xouts))

  # return
  invisible()
}

# ---------------------------------
#  function : tst.extract
# ---------------------------------

tst.extract <- function (xemfile,
                         dbug)
{
  DBUG <<- dbug
  message("  * * * * * * * * * *\n  test : extract test (currently hollow) commencing / DBUG = ", DBUG)
  message("  test : extract complete")
}

# ---------------------------------
#  junk
# ---------------------------------

# for specifier 'g' the '.precision' indicates
# significant digits, with rounding performed if
# necessary
#
# mean      <- sprintf("%.4g", mean(y))

#  $Id: extract.R 5089 2010-08-16 17:51:55Z robbie $
#  end of file
```

```
#  file-purpose     : process and display graphviz data
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Mon 15-Jun-2009 06:11 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 4050 $
#  $Date: 2009-12-11 23:36:12 +0100 (Fri, 11 Dec 2009) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/graphviz.R $

#  GRAPHVIZ
#
#  This code produces a .dot file which is subsequently
#  processed using the 'dot' utility -- using, for
#  instance:
#
#      $ dot -Tpng -o digraph.png digraph.dot
#
#  Tested with graphviz version 2.22.2 (20090313.1817).
#
#  REFERENCE
#
#  Gansner, Emden R, Eleftherios Koutsofios, and
#  Stephen C North.  2006.  Drawing graphs with 'dot'.
#  Report.  (probably published by AT&T).
#
#      download name: 'dotguide.pdf'
#      rename: 'graphviz-dot-guide.pdf'

# ---------------------------------
#  function : xem.makeGvEdge
# ---------------------------------
#  description  : makes graphviz edges
#  role         : part of the build chain
#  techniques   : text processing
#  status       : mostly complete
# ---------------------------------
#
#  the leading space on the label string is for aesthetic reasons
#
#  null heads, those with 'head' == 0, are ignored
#
# ---------------------------------

edges <- character()

xem.makeGvEdge <- function (conex,           # data structure
                            assoc,           # association index
                            tail,
                            head)
{
  # for my system (sojus), the following colorway worked well
  spectrum <- xem.colors

  # grab class details
  outclass <- "(not specified)"              # tail entity class (later grouping)
  inclass  <- "(not specified)"              # head entity class (later grouping)
  if ( tail != 0 ) outclass <- conex[[tail]][["class"]]
  if ( head != 0 ) inclass  <- conex[[head]][["class"]]

  # process color
  tecs <- "^Teas|^Junc|^Node"
  if ( TRUE                                                ) acolor <- spectrum[ 1]
  if ( charmatch("Cx"      , outclass, nomatch = FALSE) ) acolor <- spectrum[10]
  if ( charmatch("Cx"      ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 9]
  if ( charmatch("Cm"      ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 8]
# if ( charmatch("Teas"    , outclass, nomatch = FALSE)
#      &&
#      charmatch("Teas"    ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 7]
  if ( length(grep(tecs, outclass))
       &&
      length(grep(tecs, inclass)) )                        acolor <- spectrum[ 7]
  if ((charmatch("Gate"    ,  inclass, nomatch = FALSE))
       ||
      (charmatch("Gate"    , outclass, nomatch = FALSE)
       &&
       length(grep(tecs, inclass)) ))                      acolor <- spectrum[ 6]
#      charmatch("Teas"    ,  inclass, nomatch = FALSE))) acolor <- spectrum[ 6]
  if ( charmatch("Asop"    , outclass, nomatch = FALSE) ) acolor <- spectrum[ 5]
```

```
  if ( charmatch("Asop"    ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 4]
  if ( charmatch("Domain"  , outclass, nomatch = FALSE)
       &&
       charmatch("Gate"    ,  inclass, nomatch = FALSE) ) acolor <- spectrum[ 3]
  if ( charmatch("Overseer", outclass, nomatch = FALSE) ) acolor <- spectrum[ 2]

  if ( head == 0 ) acolor <- spectrum[1]
  if ( tail == 0 ) acolor <- "black"         # quite noticeable

  # create line
  line <- ""
  if ( head > 0 )                            # ignore null heads
    {
      # 02 -> 04 [label = ' 05', color = orange]
      line    <- sprintf("%02d -> %02d [label = ' %02d', color = %s]", tail, head, assoc, acolor)
      edges  <<- append(edges, line)
    }
  invisible(line)
}

# ---------------------------------
#  function : xem.makeGvNode
# ---------------------------------
#  description  : makes graphviz nodes
#  role         : part of the build chain
#  techniques   : text processing
#  status       : mostly complete
# ---------------------------------

nodes <- character()
ranks <- character(7)

xem.makeGvNode <- function (vertex,
                            i,
                            name)
{
  # make node (can remove second 'i' to omit count on label)
  # center dot: high ASCII         : \xb7     (not acceptable to UTF-8 under inkscape)
  #             XML decimal coding : &#8226;
  line   <- sprintf("%02d [label = '%02d &#8226; %s']", i, i, name)
  nodes <<- append(nodes, line)

  # determine ranking
  class <- vertex[["class"]]
  if      ( charmatch("Overseer", class, nomatch = FALSE) ) rank <- 1
  else if ( charmatch("Domain"  , class, nomatch = FALSE) ) rank <- 2
  else if ( charmatch("Asop"    , class, nomatch = FALSE) ) rank <- 3
  else if ( charmatch("Gate"    , class, nomatch = FALSE) ) rank <- 4
  else if ( charmatch("Teas"    , class, nomatch = FALSE) ) rank <- 5
  else if ( charmatch("Junc"    , class, nomatch = FALSE) ) rank <- 5
  else if ( charmatch("Node"    , class, nomatch = FALSE) ) rank <- 5
  else if ( charmatch("Cm"      , class, nomatch = FALSE) ) rank <- 6
  else if ( charmatch("Cx"      , class, nomatch = FALSE) ) rank <- 7
  else                                                      rank <- NA

  # add rank data
  if ( ! is.na(rank) ) ranks[rank] <<- sprintf("%s %02d", ranks[rank], i)
}

# ---------------------------------
#  function : xem.makeGvGraph
# ---------------------------------
#  description  : makes a graphviz graph
#  role         : part of the build chain
#  techniques   : text processing
#  status       : mostly complete
# ---------------------------------
#
#  see r2885 for code without the left-side layline
#
#  fonts: it may be necessary to change from the default
#  font "Times Roman,serif" which was not known to
#  'ImageMagick' or 'inkscape' on my system
#
# ---------------------------------

xem.makeGvGraph <- function (name = "")      # the name of the dot-file graph object (but otherwise ⬈
not used)
{
  # ---------------------------------
  #  appearance settings
```

```
  # ---------------------------------

# set <- list(over = TRUE, pts = 9, high = 0.5, font = "DejaVu Sans")  # loose (sojus)
  set <- list(over = TRUE, pts = 8, high = 0.3, font = "DejaVu Sans")  # normal(sojus)
# set <- list(over = TRUE, pts = 7, high = 0.0, font = "DejaVu Sans")  # tight (sojus)

  override   <- set$over                    # if FALSE, the remaining settings are ignored
  points     <- set$pts                     # default = 14
  nodeheight <- set$high                    # default = 0.5
  myfont     <- set$font                    # "" results in the default font "Times Roman,serif"

  # ---------------------------------
  #  attributes
  # ---------------------------------

  # create an introductory block, ## = technical issues

  intro <- character()
  if ( override ) {
  # intro <- append(intro,       "size = \"6.30, 10.63\"  // maximum size before rescaling: 160 wide x ⬋
270mm")
  # intro <- append(intro,       "margin = \"1.00, 1.00\"  // transparent")
 ## intro <- append(intro, paste("font = \"", myfont, "\""))

    intro <- append(intro,       "node [shape = box]")
    intro <- append(intro, paste("node [height = ", nodeheight,"]", sep = ""))
  # intro <- append(intro,       "node [style = filled  fillcolor = lightgray  color = lightgray]")
    intro <- append(intro,       "node [color = slategray]")
    intro <- append(intro, paste("node [fontname = \"", myfont, "\"]", sep = ""))
    intro <- append(intro,       "node [fontcolor = darkslategray]")
    intro <- append(intro, paste("node [fontsize = ", points, "]", sep = ""))

    intro <- append(intro,       "edge [color = orange]")
  # intro <- append(intro,       "edge [arrowsize = 1.0]")
  # intro <- append(intro,       "edge [style = bold]")
  # intro <- append(intro,       "edge [labelfloat = 1]")
  # intro <- append(intro,       "edge [decorate = 1]")
    intro <- append(intro, paste("edge [fontname = \"", myfont, "\"]", sep = ""))
    intro <- append(intro,       "edge [fontcolor = darkslategray]")
    intro <- append(intro, paste("edge [fontsize = ", points, "]", sep = ""))
  }
  else {
    intro <- "// using Graphviz default attributes"
  }

  # ---------------------------------
  #  layline
  # ---------------------------------

  # these labels can be easily modified
  labels    <- c("system coordination",
                 "domains",
                 "operators",
                 "gateways",
                 "assets",
                 "commodities",
                 "contexts")

  layers    <- c("overseer",
                 "domains",
                 "operators",
                 "gates",
                 "assets",
                 "commodities",
                 "contexts")

  laylines  <- character()
  laylabels <- character()
  layline   <- paste(layers, collapse = " -> ")
  for ( i in 1:length(layers) ) {
    line      <- paste(sprintf("%-11s", layers[i]), " ", "[label = '", labels[i], "']", sep = "")
    laylabels <- append(laylabels, line)
  }

  laylines <- append(laylines, "{")
  laylines <- append(laylines, "// preamble")
  laylines <- append(laylines, "node [shape = plaintext]")
  laylines <- append(laylines, "edge [style = invis]")
  laylines <- append(laylines, "// labels")
  laylines <- append(laylines, laylabels)
  laylines <- append(laylines, "// chain")
```

```
  laylines <- append(laylines, layline)
  laylines <- append(laylines, "}")

  # locally indent code
  for ( i in 2:(length(laylines) - 1) ) {
    laylines[i] <- paste("    ", laylines[i], sep = "")
  }

  # ---------------------------------
  #  rank (or layer) info
  # ---------------------------------

  rankset <- character()
  for ( i in 1:length(ranks) ) {
    buff    <- ranks[i]
    buff    <- xem.trim(buff)
    layer   <- layers[i]
    layer   <- sprintf("%-11s", layer)
    line    <- paste("rank = same;", layer, buff)
    line    <- xem.trim(line)
    line    <- paste("{", line, "}", sep = "")
    rankset <- append(rankset, line)
  }

  # ---------------------------------
  #  create output
  # ---------------------------------

  lines <- character()
  lines <- append(lines, paste("digraph '", name, "' {", sep = ""))
  lines <- append(lines, paste("// DOT file generated from:", xemfile))
  lines <- append(lines,       "// preamble")
  lines <- append(lines,       intro)
  lines <- append(lines,       "// layline (the left-side)")
  lines <- append(lines,       laylines)
  lines <- append(lines,       "// rank sets")
  lines <- append(lines,       rankset)
  lines <- append(lines,       "// node details")
  lines <- append(lines,       nodes)
  lines <- append(lines,       "// edge details (any leading space is for aesthetics)")
  lines <- append(lines,       edges)
  lines <- append(lines, "}")

  # indent code
  for ( i in 2:(length(lines) - 1) ) {
    lines[i] <- paste("    ", lines[i], sep = "")
  }

  # ---------------------------------
  #  finalize output
  # ---------------------------------

  # stringify
  linestr <- paste(lines, sep = "", collapse = "\n")   # insert newlines
  linestr <- paste(linestr, "\n", sep = "")            # add trailing newline

  # substitute ''' for '"' (to fix an earlier problem from 'sprintf')
  linestr <- gsub("'", "\"", linestr, fixed = TRUE)

  # return
  return(linestr)
}

# ---------------------------------
#  function : xem.gvTraverse
# ---------------------------------
#  description  : main conex traversal
#  role         : called by 'xem.graphviz'
#  takes        : 'conex'
#  returns      : edge call count
#  techniques   : nested 'for', 'sort', 'rev'
#  status       : complete
# ---------------------------------

xem.gvTraverse <- function (conex)
{
  calls  <- 0                                # make call count (excludes empty association lists)
  acount <- 0                                # association count (includes empty association lists)
  for ( i in 1:length(conex) )
    {
      vertex <- conex[[i]]                   # grab out vertex
```

```
      xem.makeGvNode(vertex,
                     i,
                     names(conex)[i])

      if ( length(vertex) < 2 ) {            # data preparation problem: vertex lacks adjacency list
        next                                 # applies to innermost loop
      }

      for ( j in 2:length(vertex) )
        {
          adjlist <- vertex[[j]]             # graph theory list, numeric R vector
          acount  <- acount + 1
          if ( is.null(adjlist) ) {          # necessary protection against data 'c()'
            next                             # applies to innermost loop
          }

          adjlist <- rev(sort(adjlist))      # to keep plot layering correct

          for ( k in 1:length(adjlist)  )
            {
              xem.makeGvEdge(conex,
                             acount,         # a number (or alternatively use 'names(vertex)[j]')
                             i,
                             adjlist[k])

              calls <- calls + 1
            }
        }
    }

  # return
  invisible(calls)
}

# ---------------------------------
#  function : xem.displayPNG.1
# ---------------------------------
#  description  : display PNG image
#  role         : display Graphviz representation
#  takes        : 'pngname' filename (assumed to exist)
#  returns      : TRUE on success
#  techniques   : 'display'
#  status       : complete
# ---------------------------------

xem.displayPNG.1 <- function (pngname,       # PNG file to display
                              frametitle)    # frame title string
{
  # reporting
  message("  info : xem.displayPNG.1 : commencing")
  on.exit(message("  info : displayPNG.1 : complete (on.exit)"))

  # check for presence of 'display' (call also echos the full path)
  if ( system("which display") != 0 ) {
    warning("required ImageMagick utility 'display' not found (omit job 16)")
  }

  # call creation
  call <- paste("display", "-title", frametitle, pngname, "&")

  # system call
  if ( system(call) != 0 ) {
    message("  warn : xem.displayPNG : display PNG using ImageMagick 'display' failed")
    warning("display PNG failed")
    invisible(FALSE)
  }

  # return
  invisible(TRUE)
}

# ---------------------------------
#  function : xem.displayPNG.2
# ---------------------------------
#  description  : plot PNG image
#  role         : display Graphviz representation
#  takes        : 'pngname' filename (assumed to exist)
#  returns      : TRUE on success
#  techniques   : 'EBImage-package'
#  status       : pseudo-code, requires library
# ---------------------------------
```

```
xem.displayPNG.2 <- function (pngname,            # PNG file to display
                              frametitle = NULL)  # frame title string (not used)
{
  # reporting
  message("  info : xem.displayPNG.2 : commencing")
  on.exit(message("  info : displayPNG.2 : complete (on.exit)"))

  # code
  if ( require(EBImage) ) {
    p   <- system.file("images", pngname, package = "EBImage")
    png <- readImage(p)
    display(png)
    invisible(TRUE)
  }
  else {
    message("  warn : xem.displayPNG.2 : required package 'EBImage' not present (change code to use ⬋
xem.displayPNG.1)")
    warning("required package 'EBImage' not present")
    invisible(FALSE)
  }
}

# ---------------------------------
#  function : xem.gvOverall
# ---------------------------------
#  description  : processes and displays the 'dot' dataset
#  role         : external calls
#  takes        : 'dot' DOT language string vector
#  returns      : TRUE on success
#  techniques   : 'system'
#  status       : mostly complete
#
#  typical calls
#
#      $ dot -Tpng -o my.png my.gv
#      $ display -title "new title" my.png
#
#  another possibility
#
#      $ inkview my.svg
#
# ---------------------------------

xem.gvOverall <- function (dot)
{
  # reporting
  message("  info : xem.graphviz : commencing")
  on.exit(message("  info : xem.graphviz : complete (on.exit)"))

  # check for presence of 'dot' (call also echos the full path)
  if ( system("which dot") != 0 ) {
    warning("required Graphviz utility 'dot' not found (omit job 16)")
  }

  # filenames (any ".guard" has been stripped by now)
  tag <- ".gv"                              # can be ""
  tag <- ".viz"                             # added later!
  xdotname <<- sub("\\.xem$", paste(tag, ".dot", sep = ""), xemfile)
  xpngname <<- sub("\\.xem$", paste(tag, ".png", sep = ""), xemfile)
  xsvgname <<- sub("\\.xem$", paste(tag, ".svg", sep = ""), xemfile)

  # display title
  modelno <-  sub("\\.xem$", "", xemfile)
  pos     <- regexpr("[[:digit:]][[:digit:]]$", modelno)
  modelno <- substring(modelno, pos[1])      # assumes $ is used
# buggy sample: modelno <- substr(modelno, pos[1], pos[1] + attr(pos, "match.length")[1] - 1)
  if ( pos[1] == -1 ) modelno <- ""
  if ( length(modelno) != 0 ) modelno <- paste(modelno, ":", sep = "")
  frametitle  <- paste("\"", "graphviz: ", modelno, " ", xtitle, "\"", sep = "")
# frametitle  <- paste("\"", "graphviz: ", xtitle, "\"", sep = "")

  # call creation
  call1 <- paste("dot", "-Tpng", "-o", xpngname, xdotname)
  call2 <- paste("dot", "-Tsvg", "-o", xsvgname, xdotname)

  # create 'dot' file
  cat(dot, file = xdotname)                 # write the 'dot' file

  # file processing (CAUTION: note the early returns)
  if ( system(call1) != 0 ) {                           # create PNG
```

```
    message("  warn : xem.graphviz : create PNG from dot file failed, filename ", xpngname)
    warning("create PNG failed ", xpngname)
    return(FALSE)
  }
  else if ( ! xem.displayPNG.1(xpngname, frametitle) ) {    # display PNG
    message("  warn : xem.graphviz : display PNG failed, filename ", xpngname)
  }
  else if ( system(call2) != 0 ) {                          # create SVG
    message("  warn : xem.graphviz : create SVG from dot file failed, filename ", xsvgname)
    warning("create SVG failed ", xsvgname)
    return(FALSE)
  }

  # housekeeping
  message ("  info : xem.graphviz : dot, PNG, SVG files created")

  # return success
  invisible(TRUE)
}

# ---------------------------------
#  function : xem.graphviz
# ---------------------------------
#  description  : wrapper to the core calls
#  role         : job 16 point of contact
#  takes        :
#  returns      : TRUE on success (including 'system' calls)
#  techniques   : see core calls
#  status       : incomplete
# ---------------------------------

xem.graphviz <- function (recset,
                          title)
{
  # reporting
  message("  info : xem.graphviz : commencing")
  on.exit(message("  info : xem.graphviz : complete (on.exit)"))

  # code
  conex1    <- xem.createBreadset.1(recset)
  conex2    <- xem.createBreadset.2(conex1)
  edgecalls <- xem.gvTraverse(conex2)
  dot       <- xem.makeGvGraph(xtitle)
  ret       <- xem.gvOverall(dot)

  # return
  invisible(ret)
}

# ---------------------------------
#  junk
# ---------------------------------
#
#  'sojus' command-line limited to 98304 chars,
#  which limits this call to about 1000 nodes -- the
#  solution is to write to a temporary file
#
#  dotstr <- paste("'", dot, "'", sep = "")
#  call1 <<- paste("echo", dotstr, "|", "dot", "-Tpng", "-o", xpngname)
#
#  better to use "cat("...", file = "| cmd")

#  $Id: graphviz.R 4050 2009-12-11 22:36:12Z robbie $
#  end of file
```

```
#  file-purpose     : run 'xeona' to generate or refresh data
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Tue 09-Jun-2009 16:23 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 5082 $
#  $Date: 2010-08-16 16:10:00 +0200 (Mon, 16 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/run.R $

# ---------------------------------
#  function : xem.xemconf.model
# ---------------------------------
#  description  : simply checks for existence
#  role         : preparation
#  takes        : 'xemname' string
#  returns      : 'xemname' string
#  techniques   : 'file.exists'
#  status       : complete
# ---------------------------------

xem.xemconf.model <- function (xemname)
{
  # reporting
  message("  info : xem.xemconf.model : commencing")
  on.exit(message("  info : xem.xemconf.model : complete (on.exit)"))

  # report
  if ( ! file.exists(xemname) ) {
    message("  warn : xem.xemconf.model : xem file not present: ", xemname)
    warning("xem file not found ", xemname)
  }

  # return
  return(xemname)
}

# ---------------------------------
#  function : xem.xemconf.guard
# ---------------------------------
#  description  : cleans model name and checks for existence
#  role         : preparation
#  takes        : 'xemname' string
#  returns      : 'xemname' string duly cleaned
#  techniques   : 'sub', 'file.exists'
#  status       : complete
#
#  note that this program is currently limited
#  models with guard files
#
# ---------------------------------

xem.xemconf.guard <- function (xemname)
{
  # reporting
  message("  info : xem.xemconf.guard : commencing")
  on.exit(message("  info : xem.xemconf.guard : complete (on.exit)"))

  # process names
  xemname  <- sub("(|\\.xem)$"        ,       ".xem", xemname)   # add ".xem" extension if absent
  xemguard <- sub("(|\\.guard)\\.xem$", ".guard.xem", xemname)   # guard file name if not already
  xemmodel <- sub("\\.guard\\.xem$"   ,       ".xem", xemname)   # normal model file

  # report
  if ( ! file.exists(xemmodel)  ) {
    message("  info : xem.xemconf.guard : xem model file not present: ", xemmodel)
  }
  if ( ! file.exists(xemguard) ) {
    message("  warn : xem.xemconf.guard : xem guard file not present: ", xemguard)
    warning("xem guard file not found ", xemguard)
  }

  # return
  return(xemmodel)
}

# ---------------------------------
#  function : xem.xeona
# ---------------------------------
```

```
#  description  : invokes xeona
#  role         : a step in the call chain
#  takes        : 'xeona', 'xemname', optionally 'mode', optionally 'report' reporting level
#  returns      : 'xret' (this is buggy, see caution below)
#  side-effects : sets 'xcall'
#  techniques   : 'system'
#  status       : complete
#
#  CAUTION: some internal problem with non-zero system
#  return values, for instance ( system("false") )
#  yields 256 and not 1 as expected
#
#  CAUTION: cannot add 'grep' filters to the call
#  easily, without working tricks with the bash
#  'PIPESTATUS' builtin -- either that or run a
#  successful call twice
#
# ---------------------------------

xem.xeona <- function (xeona,
                       xemname,
                       mode     = 7,
                       report   = 0,
                       exittrip = 1,
                       nodata   = FALSE)
{
  # reporting
  message("  info : xem.xeona : commencing")
  on.exit(message("  info : xem.xeona : complete (on.exit), xret ", xret))

  # preparation
# xemname <- sub("\\.xem$", "", xemname)     # trim trailing ".xem", strictly optiona1

  # preamble
  xopt  <- character()
  xopt  <- append(xopt, paste("--guard"))
# xopt  <- append(xopt, paste("--pepper"))   # leads to testing instabilities
  xopt  <- append(xopt, paste("--mode"    , mode))
  xopt  <- append(xopt, paste("--report"  , report))
  xopt  <- append(xopt, paste("--exittrip", exittrip))
  if ( nodata ) xopt <- append(xopt, paste("--nodata"))
  xopt  <- append(xopt, paste("--file", xemname))
  xopts <- paste(xopt, collapse = "  ")      # double space is correct
  xcal  <- paste(xeona, xopts, sep = "  ")   # double space is correct
  message("xeona call     : ", xcal)

  # rule
  rule <- paste("  ", paste(rep("=", 90), collapse = ""), "\n",sep = "")

  # return value
  xret <- NA                              # should be overwritten

  # system call
  if ( report == 0 ) {
    redir <- "1>/dev/null 2>/dev/null"
    call  <- paste(xcal, redir, sep = " ")
    xret  <- system(call, intern = FALSE, ignore.stderr = FALSE)
  }
  else if ( report < 3 ) {
    cat(rule)
    redir <- "1>/dev/null"
    call  <- paste(xcal, redir, sep = " ")
    xret  <- system(call, intern = FALSE, ignore.stderr = FALSE)
    cat(rule)
  }
  else {
    cat(rule)
    call  <- paste(xcal, sep = " ")
    xret  <- system(call, intern = FALSE, ignore.stderr = FALSE)
    cat(rule)
  }

  # set the exported call string
  xcall  <<- xcal

  # return
  return(xret)
}

# ---------------------------------
#  function : xem.freshen
```

```
# ---------------------------------
#  description  : run xeona binary on XEM file in guard mode
#  role         : ensure the nominated XEM file is fresh
#  takes        : 'xeona' binary, 'xemname' filename, 'report' reporting level
#  returns      : best achieved mode (7 thru 0, 7 is ideal, 0 means even mode 1 failed)
#  techniques   : 'super-assignment operator'
#  status       : complete
#
#  the various modes are described in 'xeona' option '--usage'
#
#  the returned 'mode' variable may be used to control
#  future actions, for instance, timeseries from mode 3
#  and less are unlikely to work -- this variable is
#  also used for labeling plots
#
# ---------------------------------

xem.freshen <- function (xeona,              # binary name
                         xemname,            # XEM model name
                         report = 0)
{
  # reporting
  message("  info : xem.freshen : commencing")
  on.exit(message("  info : xem.freshen : complete with mode (on.exit) ", mode))

  # get some meta info
  xemguard <- sub("(|\\.guard)\\.xem$", ".guard.xem", xemname)
  data     <- xem.slurp(xemguard)            # temporary slurp
  xtrip   <<- as.numeric(xem.scanForInValue(data, ↙
"program.run-script-settings.script-option-exittrip"))
  xnodata <<- as.logical(as.numeric(xem.scanForInValue(data, ↙
"program.run-script-settings.script-option-nodata")))

  # step backward thru modes
  modes <- c(7, 6, 5, 4, 3, 2, 1)            # deep to shallow
  for ( mode in modes ) {
    ret <- xem.xeona(xeona, xemname, mode, report, xtrip, xnodata)
    if ( ret == 0 ) break                    # success
    mode <- 0                                # mode 0 indicates that even mode 1 failed
  }

  # abandon on mode zero
  if ( mode == 0 ) {
    stop("modes ", modes ," failed (the xem file may be missing)")
  }

  # return
  return(mode)
}

# ---------------------------------
#  function : xem.slurp
# ---------------------------------
#  description  : open file and read in XEM file
#  role         : obtain data
#  takes        : 'xemname' filename
#  returns      : read-in line-oriented data
#  techniques   : 'readLines'
#  status       : complete
# ---------------------------------

xem.slurp <- function (xemname)
{
  # reporting
  message("  info : xem.slurp : commencing with model file = ", xemname)
  on.exit(message("  info : xem.slurp : complete (on.exit)"))

  # active code
  data  <- readLines(xemname)                # default is to read all
  lines <- length(data)
  message("lines read     : ", lines)
  if ( lines == 0 ) warning("no lines read")

  # return
  invisible(data)
}

# ---------------------------------
#  function : xem.getRunPolicy
# ---------------------------------
#  description  : obtain run policy by regex scanning the data
```

```
#  role         : not used
#  takes        : read-n data
#  returns      : run policy
#  techniques   : 'grep', 'as.integer'
#  status       : mostly complete but not tested
#
#  caution: better to use 'xem.scanForValue'
#
# ---------------------------------

xem.getRunPolicy <- function (data)          # unprocessed xem data
{
  trigger <- "r-policy"
  regex   <- paste("^[[:blank:]]*", trigger, "[[:blank:]]*", ">", sep = "")
  hit     <- grep(regex, data)
  len     <- length(hit)
  if ( len == 0 ) {
    message("  ")
    ret = NA
  }
  else if ( len == 1 ) {                     # looking good ..
    policy <- strsplit(">", data[hit])[2]    # grab value
    prior  <- options(warn = WARNSET)        # negative is suppress warnings, zero is store warnings
    ret    <- as.integer(policy)             # NA on failure
    options(warn = prior$warn)               # restore warnings
    if ( is.na(ret) ) {
      message("  ")
    }
  }
  else {
    message("  ")
    ret = NA
  }
  return(ret)
}

# ---------------------------------
#  function : xem.scanForInValue
# ---------------------------------
#  description  : finds field value in unprocessed data
#  role         : any value recovery prior to loading the nested data-structure
#  takes        : slurped 'data' and fully-resolved 'fieldname'
#  returns      : sought value, else 'NULL' if not found or 'NA' if arguments fail integrity checks
#  techniques   : nested 'if' statements (ugly)
#  status       : close-to-complete
#
#  example
#
#    r.policy <- xem.scanForInValue(data,
#                                   "program.r-processing.r-policy")
#
# ---------------------------------

xem.scanForInValue <- function (data,        # unprocessed xem data
                                fieldname)   # fully-resolved field name
{
  # split 'fieldname'
  dotseps <- unlist(strsplit(fieldname, ".", fixed = TRUE))

  # integrity checks
  if ( length(dotseps) != 3 ) {              # three part split expected
    return(NA)
  }
  if ( sum(nchar(data)) == 0 ) {             # xem data character count is zero
    return(NA)
  }

  # further processing
  reckind <- dotseps[1]                      # in { program, entity, ... }
  id      <- dotseps[2]                      # trailing separator, which may or may not be hardcoded ⬀
in 'xeona'
  fname   <- dotseps[3]                      # field name

  dotsep  <- paste(dotseps, collapse = ".")  # for reporting

  # main loop
  procFlag <- FALSE                          # flag to enable further processing of the current record
  for ( i in 1:length(data) )
    {
      line <- data[i]                        # grab next line
```

```
      pat <- paste("^", reckind, ".", id, sep = "")
      if ( regexpr(pat, line) < 1                       # a hit for "^reckind.id"
           ||                                           # .. or ..
           procFlag == TRUE  )                          # the current record has been selected
        {
          if ( regexpr("^[[:alpha:]#]", line) > 0 )     # meaning the next (and possibly disabled) ⬋
record encountered
            {
              procFlag <- FALSE                         # unselect this record
            }
          else
            {
              procFlag <- TRUE                          # select this record

              inseps <- unlist(strsplit(line, ">", fixed = TRUE))

              if ( length(inseps) == 2 )                # meaning in-data encountered
                {
                  left  <- xem.trim(inseps[1])
                  right <- xem.trim(inseps[2])
                  if ( regexpr(fname, left, fixed = TRUE) > 0 )  # a hit for the field name
                    {
                      message("  info : xem.scanForValue : value encountered for in-field '", dotsep, ⬋
"' of ", right )
                      return(right)
                    }
                }
            }
        }
    }
  message("  warn : xem.scanForValue : NO value encountered for ", dotsep)
  return(NULL)
}

# ---------------------------------
#  junk
# ---------------------------------
#
#  filt <- "" #"| grep \"^[[:blank:]]*[[:digit:]]+[[:print:]]+WARN\""

#  $Id: run.R 5082 2010-08-16 14:10:00Z robbie $
#  end of file
```

```
#  file-purpose     : save open plot windows in various formats
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Mon 16-Aug-2010 18:16 UTC
#  file-status      : work-in-progress
#  file-keywords    : xeona R

#  $Revision: 5137 $
#  $Date: 2010-08-20 02:04:38 +0200 (Fri, 20 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/saveplots.R $

# ---------------------------------
#  function : xem.rofile
# ---------------------------------
#  description  : make file read-only
#  role         : normally called after file creation
#  takes        : 'filename' as string
#  returns      : TRUE on success
#  techniques   : 'system' 'chmod'
#  status       : complete
#  CAUTION      : Linux-specific ('chmod')
# ---------------------------------

xem.rofile <- function (filename,
                        kill.empty = TRUE)
{
  if ( ! is.logical(kill.empty) ) {
    stop("expecting a logical argument type for 'kill.empty'")
  }

  if ( length(filename) !=  1 ) {
    if ( nchar(filename) == 0 ) {
      message("  warn : xem.rofile : problem filename ", filename)
      warning("problem filename")
      return(FALSE)
    }
  }

  if ( ! file.exists(filename) ) {
    message("  warn : xem.rofile : cannot locate file ", filename)
    warning("cannot locate file ", filename)
    return(FALSE)
  }

  # the following CANNOT be implemented using 'Sys.chmod'
  modestr <- paste("a-w", "o-r", "ug+r", sep = ",")
  ret <- system(paste("chmod", "--silent", modestr, filename, sep = " "))

  if ( kill.empty ) {                       # logical
    finfo <- file.info(filename)            # data frame
    size  <- finfo[ , "size"]               # CAUTION: the comma is correct, yields a numeric
    if ( size == 0 ) file.remove(filename)
  }

  if ( ret == 0 ) TRUE else FALSE
}

# ---------------------------------
#  function : xem.saveplot.png
# ---------------------------------
#  description  : save given device as PNG
#  status       : working
#
#  R version    : 2.10.1 (2009-12-14)
#  development  : Ubuntu 10.04 Linux 2.6.32-23-generic / Intel Core i5 laptop /'hinau'
#  see also     : bash script 'x11show.sh'
# ---------------------------------

# [1] 700 is too tight, 1100 and 1200 are okay, 1700 is rather loose

xem.saveplot.png <- function (device,
                              filestub,             # save name
                              type,                 # plot type
                              dbug,
                              width.px  = 1200)     # bitmap width [1]
{
  # set debug level here
  DBUG <- dbug

  # report
```

```
  message("  info : xem.saveplot.png : commencing with",
          " device = "      , device,
          " / filestub = '" , filestub, "'",
          " / type = '"     , type, "'",
          " / dbug = "      , dbug,
          " / width.px = "  , width.px)
  on.exit(message("  info : xem.saveplot.png : complete (on.exit)"))

  # print calls - no dimensions specified

  xem.saveplot.printme(png, filestub, type, "a", width.px)

  # clean run indication
  message("  info : xem.saveplot.png : complete run")

}

# ---------------------------------
#  function : xem.saveplot.pdf
# ---------------------------------
#  description  : save given device as PDF
#  status       : work-in-progress
#
#  R version    : 2.10.1 (2009-12-14)
#  development  : Ubuntu 10.04 Linux 2.6.32-23-generic / Intel Core i5 laptop 'hinau'
#  see also     : bash script 'x11show.sh'
#
#  CAUTION: coded on the assumption 'cairo' is supported
#
#  CAUTION: 'xeona' breadboard plots are sometimes scrambled
#           with some settings -- this can happen across all
#           vector graphics plots at least
# ---------------------------------

# [1] 150mm is scrambled, 300mm is okay

xem.saveplot.pdf <- function (device,
                              filestub,             # save name
                              type,                 # plot type
                              dbug,
                              width.mm   = 300,     # vector width for format b [1]
                              doctitle   = "",      # CAUTION: NULL produces NA
                              format.alt = TRUE)    # use alternative formats
{
  # set debug level here
  DBUG <- dbug

  # report
  message("  info : xem.saveplot.pdf : commencing with",
          " device = "      , device,
          " / filestub = '" , filestub, "'",
          " / type = '"     , type, "'",
          " / dbug = "      , dbug,
          " / width.mm = "  , width.mm,
          " / doctitle = '" , doctitle, "'",
          " / format.alt =" , format.alt)

  ##  TOFIX: 17-Aug-2010: solve or delete
  ##  CALL <- match.call()
  ##  message("  info : xem.saveplot.pdf : call = ", CALL) # TOFIX: 17-Aug-2010: experimental

  on.exit(message("  info : xem.saveplot.pdf : complete (on.exit)"))

  # report on cairo
  if ( DBUG > 0 ) {
    x <- capabilities("cairo")
    if ( length(x) > 0 )
      if ( x )
        message("  info : xem.saveplot.pdf : this R installation supports cairo")
      else
        message("  info : xem.saveplot.pdf : this R installation does NOT support cairo")
  }

  # R device    : cairo_pdf
  # creator     : cairo 1.8.10
  # fonts       : NimbusSansL (Type 1) / embedded
  # PDF version : 1.4

  # print calls - no dimensions specified

  xem.saveplot.printme(cairo_pdf, filestub, type, "a")
```

```
  if ( format.alt ) {
    xem.saveplot.printme(cairo_pdf, filestub, type, "d", width.mm, 1.00)
  }

  # clean run indication
  message("  info : xem.saveplot.pdf : complete run")

}

# ---------------------------------
#  function : xem.saveplot.svg
# ---------------------------------
#  description  : save given device as SVG
#  status       : work-in-progress
#
#  R version    : 2.10.1 (2009-12-14)
#  development  : Ubuntu 10.04 Linux 2.6.32-23-generic / Intel Core i5 laptop /'hinau'
#
#  CAUTION: 'xeona' breadboard plots are sometimes scrambled
#           with some settings -- this can happen across all
#           vector graphics plots at least
# ---------------------------------

# [1] 150mm is scrambled, 200mm is over-tight, 300mm is okay

xem.saveplot.svg <- function (device,
                              filestub,             # save name
                              type,                 # plot type
                              dbug,
                              width.mm   = 300,     # vector width for alternative format [1]
                              doctitle   = "",      # CAUTION: NULL produces NA
                              format.alt = TRUE)    # use alternative formats
{
  # set debug level here
  DBUG <- dbug

  # report
  message("  info : xem.saveplot.svg : commencing with",
          " device = "      , device,
          " / filestub = '" , filestub, "'",
          " / type = '"     , type, "'",
          " / dbug = "      , dbug,
          " / width.mm = "  , width.mm,
          " / doctitle = '" , doctitle, "'",
          " / format.alt =" , format.alt)

  on.exit(message("  info : xem.saveplot.svg : complete (on.exit)"))

  # XML version  : 1.0
  # encoding     : UTF-8
  # fonts        : no fonts, rather characters are cloned to vectored glyphs defined in <defs>
  # default size : width="850pt" height="424pt

  # 'title' argument not supported
  # 'width' alone is okay

  # print calls - no dimensions specified

  xem.saveplot.printme(svg, filestub, type, "a")

  if ( format.alt ) {
    scale <- 0.67

    xem.saveplot.printme(svg, filestub, type, "b1", scale * width.mm, 0.50)
    xem.saveplot.printme(svg, filestub, type, "c1", scale * width.mm, 0.67)
    xem.saveplot.printme(svg, filestub, type, "d1", scale * width.mm, 1.00)
    xem.saveplot.printme(svg, filestub, type, "e1", scale * width.mm, 1.41)
    xem.saveplot.printme(svg, filestub, type, "f1", scale * width.mm, 2.00)

    scale <- 1.00

    xem.saveplot.printme(svg, filestub, type, "b2", scale * width.mm, 0.50)
    xem.saveplot.printme(svg, filestub, type, "c2", scale * width.mm, 0.67)
    xem.saveplot.printme(svg, filestub, type, "d2", scale * width.mm, 1.00)
    xem.saveplot.printme(svg, filestub, type, "e2", scale * width.mm, 1.41)
    xem.saveplot.printme(svg, filestub, type, "f2", scale * width.mm, 2.00)

    scale <- 1.41

    xem.saveplot.printme(svg, filestub, type, "b3", scale * width.mm, 0.50)
    xem.saveplot.printme(svg, filestub, type, "c3", scale * width.mm, 0.67)
```

```
    xem.saveplot.printme(svg, filestub, type, "d3", scale * width.mm, 1.00)
    xem.saveplot.printme(svg, filestub, type, "e3", scale * width.mm, 1.41)
    xem.saveplot.printme(svg, filestub, type, "f3", scale * width.mm, 2.00)
  }

  # clean run indication
  message("  info : xem.saveplot.svg : complete run")

}

# ---------------------------------
#  function : xem.saveplot.printme
# ---------------------------------
#  description  :
#  role         :
#  takes        :
#  returns      :
#  techniques   :
#  status       : under testing
#
#  typical call
#
#    xem.saveplot.printme(svg, filestub, type, "d", width.mm, 2.000)
#
#  infeasible layout (solution change
#
# Error in dev.copy(device = function (filename = if (onefile) "Rplots.svg" else "Rplot%03d.svg",  :
#   invalid graphics state
#
# the plot shows: "figure margins too large"
#
# ---------------------------------

xem.saveplot.printme <- function (device,         # { png cairo_pdf svg}
                                  filestub,       # filestub
                                  type,           # { summary .. }
                                  tag,            # { a b c .. }
                                  width  = NA,    # px or mm
                                  ratio  = NA)    # 1.000 gives a square
{
  # report
  message("  info : xem.saveplot.printme : commencing with",
          " device = "      , "(not recordable)",
          " / filestub = '" , filestub, "'",
          " / tag = '"      , tag, "'",
          " / width = "     , width,
          " / ratio = "     , ratio)
  on.exit(message("  info : xem.saveplot.printme : complete (on.exit)"))

  # obtain format information
  devname <- deparse(substitute(device))     # based on code from 'dev.print'
  if      ( devname == "png"      ) extn <- "png"
  else if ( devname == "cairo_pdf") extn <- "pdf"
  else if ( devname == "svg"      ) extn <- "svg"
  else stop("unsupported device", devname)

  # process variables
  filename  <- paste(filestub, type, tag, extn, sep = ".")  # say "filestub.summary.a.svg"
  reportstr <- paste(toupper(extn), tag, sep = " ")         # say "SVG b"

  # print calls
  printme.0 <- function()
    {
      # background color not shown by 'display', but described in "Image Info"
      message("  info : xem.saveplot.printme : making 0: ", filename)
      if ( file.exists(filename) ) file.remove(filename)
      from <- dev.print(device = device,
                        file   = filename,
                        width  = width,
                        bg     = "white")    # CAUTION: "transparent" would not take
      xem.rofile(filename)
      xem.report(filename, reportstr, from, width)
    }

  printme.1 <- function()
    {
      # 'title' argument not supported for pdf
      message("  info : xem.saveplot.printme : making 1: ", filename)
      if ( file.exists(filename) ) file.remove(filename)
      from <- dev.print(device = device,
                        file   = filename)
```

```
      xem.rofile(filename)
      xem.report(filename, reportstr, from)
    }

  printme.2 <- function()
    {
      message("  info : xem.saveplot.printme : making 2: ", filename)
      if ( file.exists(filename) ) file.remove(filename)
      from <- dev.print(device = device,
                        file   = filename,
                        width  = width / 25.400,
                        height = width / 25.400 * ratio)
      xem.rofile(filename)
      xem.report(filename, reportstr, from, width, ratio)
    }

  # make appropriate call (the logic here a little subtle)
  if      ( is.na(ratio) && ! is.na(width) ) try( printme.0() )
  else if ( is.na(width)                   ) try( printme.1() )
  else                                       try( printme.2() )

  # clean run indication
  message("  info : xem.saveplot.printme : complete run")

}

# ---------------------------------
#  function : xem.report
# ---------------------------------
#  description  : assemble and submit a standard report line
#  role         : the various 'xem.saveplot.*' calls
#  takes        : see below
#  returns      : nothing of interest
#  techniques   : 'sprintf' 'append'
#  status       : under testing
#
#    the typical call:
#
# xem.report(svgfilec, "SVG c", svgc, width.mm, ratio)
#
#    gives approximately
#
# '    ../test-12.summary.c.svg               tag = SVG c   from = 2   width = 0300  ratio = 1.500'
#
# ---------------------------------

xem.report <- function (filename,            # filename as string
                        tag,                 # 5-char tag as string
                        fromdev,             # number of from device
                        width    = NA,       # width in px or mm as number
                        ratio    = NA)       # ratio as number
{
  # check for bad coding
  if ( ! is.na(ratio) && is.na(width) ) stop("if 'ratio' is used, 'width' is also required")

  # concat
  repline <- sprintf("    %-70s  tag = %-6s   from = %d", filename, tag, fromdev)
  if ( ! is.na(width) ) {
    repline <- sprintf("%s   width = %04d", repline, width)
  }
  if ( ! is.na(ratio) ) {
    repline <- sprintf("%s   ratio = %.3f", repline, ratio)
  }
  repline <- sprintf("%s\n", repline)

  # export details
  saveplot.report <<- append(saveplot.report, repline)
}

#  $Id: saveplots.R 5137 2010-08-20 00:04:38Z robbie $
#  end of file
```

```
#  file-purpose     : test suite - source directly to run
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Thu 04-Jun-2009 20:46 UTC
#  file-status      : ongoing
#  file-keywords    : xeona R

#  $Revision: 5076 $
#  $Date: 2010-08-14 18:20:11 +0200 (Sat, 14 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/test.R $

#  ADDITIONAL REQUIREMENTS
#
#    userFuncs.R    # normally loaded automatically
#
#  USAGE
#
#    > source("test.R")

# ---------------------------------
#  preamble
# ---------------------------------

message("  file : test.R : commencing")
keep <- c("keep")                             # keep me too!

keep     <- c(keep, "BROWSER", "BESTMODE")
BROWSER  <- TRUE                              # better than ask = T
BESTMODE <- 0                                 # 'xeona' best mode state, should be overwritten

opar <- par(no.readonly = TRUE)               # all par settings which could be changed
keep <- c(keep, "opar")
# par(ask = F)                                # T = prompt, F = rush past
# par(new = T)

# ---------------------------------
#  utilities test
# ---------------------------------

robbie.clean(keep)                            # CAUTION: note the clean calls in this script
source("utils.R")
tst.utils(dbug = 1)
if ( BROWSER ) browser()
plot.new()
keep <- c(keep)

# ---------------------------------
#  breadboard test
# ---------------------------------

# dbug 4 = print color test and (currently) stop

robbie.clean(keep)
source("utils.R")
source("breadplot.R")
xmode <- 0                                    # for test purposes
tst.breadboard(dbug = 1)
if ( BROWSER ) browser()
plot.new()
keep <- c(keep, "capture", "conex")           # 'conex' is hardcoded rather than generated

# ---------------------------------
#  data processing test
# ---------------------------------

# relatively hollow, test with point of entry test instead

robbie.clean(keep)
source("dataproc.R")
tst.dataproc(xemfile = "test.xem",            # hence 'test.guard.xem'
             dbug    = 1)
if ( BROWSER ) browser()
keep <- c(keep)

# ---------------------------------
#  extract test
# ---------------------------------

robbie.clean(keep)
source("extract.R")
```

```
tst.extract(dbug = 1)
if ( BROWSER ) browser()
plot.new()
keep <- c(keep)

# ---------------------------------
#  point of entry test
# ---------------------------------

robbie.clean(keep)
source(file.path(Sys.getenv("XEONAR"), "xem.R"))
xem.job(xemfile = "../xeona1/xeona-xmoks/test-context.xem",
        xeona   = "../xeona1/xeona.mach",
        mlabel  = "intentionally empty",
        dbug    = 1)
# if ( BROWSER ) browser()
# plot.new()
keep <- c(keep, "data", "policy", "recset")

# ---------------------------------
# report on lines of code
# ---------------------------------

source("utils.R")
message("  info : cat/grep/wc lines of code: ", xem.linesOfCode())

# ---------------------------------
#  housekeeping
# ---------------------------------

par(opar)
rm(list = "opar")

robbie.clean(keep, say = TRUE )              # 'keep' is the retains list

message("  file : test.R : finishing")

#  $Id: test.R 5076 2010-08-14 16:20:11Z robbie $
#  end of file
```

```
#  file-purpose     : simple trial script for 'xem.job'
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Fri 05-Jun-2009 09:17 UTC
#  file-status      : ongoing
#  file-keywords    : xeona R

#  $Revision: 5076 $
#  $Date: 2010-08-14 18:20:11 +0200 (Sat, 14 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/trial.R $

#  ADDITIONAL REQUIREMENTS
#
#    userFuncs.R    # normally loaded automatically
#
#  USAGE
#
#    > source("trial.R")

# ---------------------------------
#  commencement
# ---------------------------------

message("  file : trial.R : starting")

robbie.clean()
robbie.killX11s()

source(file.path(Sys.getenv("XEONAR"), "xem.R"))

# ---------------------------------
#  specify external files
# ---------------------------------

xeona   <- "../xeona1/xeona.mach"

xemfile <- "../xeona1/xeona-xmoks/test-context.xem"
xemfile <- "../xeona1/xeona-xmoks/experiment-01.xem"

# ---------------------------------
#  call job
# ---------------------------------

xem.job(xemfile,
        xeona,
        dbug = 0)                           # debug level in { 0, .. 4 }

# ---------------------------------
#  completion
# ---------------------------------

rm(list = c("xem.job"))
rm(list = c("xeona", "xemfile"))            # from here
# rm(list = c(""))                            # from 'xem.R'

message("  file : trial.R : finishing")

#  $Id: trial.R 5076 2010-08-14 16:20:11Z robbie $
#  end of file
```

```
#  file-purpose     : utility functions
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Thu 04-Jun-2009 19:55 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 5116 $
#  $Date: 2010-08-18 11:45:29 +0200 (Wed, 18 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/utils.R $

# ---------------------------------
#  function : xem.trim
# ---------------------------------
#  description  : trim space and/or tab characters from both ends of a given string
#  role         : utility
#  takes        : string (or vector of strings)
#  returns      : string (or vector of strings)
#  techniques   : 'sub'
#  status       : complete
#
#  terminology-wise, a normal string (in other
#  languages) is officially known as a "single
#  character string" and represented by a "length one
#  character vector"
#
# ---------------------------------

xem.trim <- function (s)
{
  if ( ! is.character(s) ) {
    warning("string vector expected")
  }
  s <- sub("^[[:blank:]]+", "", s)
  s <- sub("[[:blank:]]+$", "", s)
  return(s)
}

# ---------------------------------
#  function : xem.beep
# ---------------------------------
#  description  : predefined beep function
#  role         : utility
#  takes        : type in { alert error weird }
#  returns      : system exit
#  techniques   : 'beep'
#  status       : complete
#  CAUTION      : platform-specific (requires a 'beep')
#
#  this function requires either:
#
#    * the Linux package 'beep' - advanced pc-speaker beeper
#    * a substitute bash script, bash function, or bash alias
#
#  some troubleshooting calls:
#
#    $ which beep
#    $ type beep
#
# ---------------------------------

xem.beep <- function (type = "alert")
{
  switch(type,
         "alert" = beepcall <- "beep -f 2000 -r 1",
         "error" = beepcall <- "beep -f 300 -n -f 350 -n -f 450",
         "weird" = beepcall <- "beep -f 800 -n -f 1800 -n -f 1300 -n -f 900 -n -f 1100")
  retBeep <- system(beepcall)                # system call
}

# ---------------------------------
#  function : xem.linesOfCode
# ---------------------------------
#  description  : estimate source lines of code
#  role         : background
#  takes        : n/a
#  returns      : source lines of code
#  techniques   : 'cat' 'grep' 'wc'
#  status       : complete
# ---------------------------------
```

```
# $ cat --squeeze-blank *.R | grep --invert-match "etc" | wc --lines

xem.linesOfCode <- function ()
{
  # the escape on the '#' is to preserve emacs alignment rules
  cal <- character()
  cal <- append(cal, "cat --squeeze-blank *.R")
  cal <- append(cal, "grep --invert-match \"^[[:blank:]]*$\"")   # empty line
  cal <- append(cal, "grep --invert-match \"^[[:blank:]]*#\"")   # comment line
  cal <- append(cal, "grep --invert-match \"^[[:blank:]]*}$\"")  # single '}' line
  cal <- append(cal, "wc --lines")
  cat <- paste(cal, collapse = " | ")
  loc <- system(cat, intern = TRUE)
  loc                                       # additional statement required for visibility
}

# ---------------------------------
#  function : xem.svnver
# ---------------------------------
#  description  : recover current subversion version
#  role         : utility
#  takes        : n/a
#  returns      : svn version as integer or zero if unclean
#  techniques   : 'system' 'svnversion'
#  status       : complete
#  CAUTION      : Linux-specific (possibly)
#
#  the 'system' argument "intern = TRUE" setting is not
#  supported on all platforms -- but Linux should be
#  safe
#
#  "unclean" means uncommitted changes or out-of-sync working copy or both
#
#  CAUTION: the path '/home/robbie/xeona/svn2/futz' may
#  need adjusting.
#
# ---------------------------------

xem.svnver <- function ()
{
  message("  info : xem.svnver : calling 'svnversion' (can be slow)")
  on.exit(message("  info : xem.svnver : complete (on.exit), elapsed time ", elapsed, "s"))

  time    <- system.time(verstr <- system("svnversion /home/robbie/synk/xeona/svn2/futz", intern = ⬋
TRUE))
  elapsed <- round(time[3], 0)

  prior   <- options(warn = -1)             # negative is suppress warnings, zero is store warnings
  ver     <- as.numeric(verstr)             # string to number coercion
  options(warn = prior$warn)                # restore warnings
  if ( is.na(ver) ) ver <- 0                # reset to zero if svn version is "unclean"
  message("  info : xem.svnver : xeona version : ", ver)

  invisible(ver)                            # return invisibly
}

# ---------------------------------
#  function : xem.preamble
# ---------------------------------
#  description  : parse subversion 'Id' ident and return file name (see example)
#  role         : not used
#  takes        : 'Id' ident line
#  returns      : file name
#  techniques   : 'strsplit'
#  status       : complete
#
#  example (replace DOLLAR with $)
#
#    DOLLARId: file.R 5113 2010-08-18 08:26:13Z robbie DOLLAR
#
# ---------------------------------

xem.preamble <- function (svnid)
{
  # CAUTION: a space char is needed between the
  # double-quotes and the svn:keyword 'Id' dollar chars
  # at both ends

  svnids <- strsplit(svnid, " ", fixed = TRUE)    # the first element is duly ""
  rfile  <- unlist(svnids)[3]
```

```
  rev    <- unlist(svnids)[4]
  invisible(rfile)                                # return invisibly
}

# ---------------------------------
#  function : xem.pyramidSort
# ---------------------------------
#  description  : undertakes what I called a "pyramid sort"
#  role         : used by 'xem.breadTraverse' to keep the arrow layering correct
#  takes        : 'numbers' number sequence, 'peak' which need not be in 'numbers'
#  returns      : pyramid sort of same length
#  note         : 'peak' is included once in the middle if it matches
#  techniques   : 'is.numeric' 'sort' 'c'
#  status       : complete
#
#  example
#
#     numbers :  1 3 7 8 9 12 13
#     peak    :  8
#     output  :  1 3 7 8 13 12 9
#
#  see Crawley (2007 pp21-22) for a discussion on logical
#  subscripts within vectors
#
# ---------------------------------

xem.pyramidSort <- function(numbers,
                            peak)
{
  if ( ! ( is.numeric(numbers) && is.numeric(peak) ) ) {
    warning("numeric arguments required")
    return(numbers)
  }

  if ( length(peak) != 1 ) {
    warning("peak must be a scalar ", length(peak))
    return(numbers)
  }

  buf <- sort(numbers)
  lo  <- buf[buf <= peak]
  hi  <- buf[buf >  peak]
  buf <- c(lo, rev(hi))
}

# ---------------------------------
#  function : xem.isTwoContained
# ---------------------------------
#  description  : test a given aggregate against a pure mask
#  role         : decoding aggregates
#  takes        : a 'pure' mask > 0 and an 'aggregate' test value >= 0, both integer-valued
#  returns      : logical if 2-contained
#  techniques   : integer-style (and not bitwise) arithmetic
#  status       : complete
#
#  examples
#
#      xem.isTwoContains(13, 4) returns  TRUE, reduced vector 0 8 4 0 1
#      xem.isTwoContains(11, 4) returns FALSE, reduced vector 0 8 0 2 1
#      xem.isTwoContains(4,  8) returns FALSE, reduced vector   0 4 0 0
#      xem.isTwoContains(0,  1) returns FALSE, reduced vector         0
#      xem.isTwoContains(13)    returns vector 0 8 4 0 1
#      xem.isTwoContains( 0)    returns vector         0
#      xem.isTwoContains()      returns NA
#
#  this function uses integer arithmetic, rather than
#  more elegant bitwise arithmetic
#
#  in comparison to C/C++, integer arithmetic in R a
#  somewhat opaque -- still the following code tested
#  out okay
#
#  note the following
#
#      is.integer(4)   returns FALSE
#      is.integer(4:4) returns TRUE
#
#  note in particular the '%%' 'modulus' operator
#  (known elsewhere as the 'modulo' and/or 'remainder'
#  operator), and the inconsistent convention for
#  negative inputs (although not used here) (see
```

```
#  wikipedia for details)
#
#  note also the 'bitopts' add-on package from CRAN --
#  probably useful if a bitwise rewrite is indicated
#
#    bitops - functions for bitwise operations on
#             integer vectors
#
#  modeled on 'xeona' C++ code for 'xeona::isTwoContained'
#  -- note the argument order is reversed
#
# ---------------------------------

xem.isTwoContained <- function (aggregate = 0,    # integer under investigation (0 or greater)
                                pure      = 1)    # mask as power of two (1 or greater)
{
  # return if 'aggregate' (and probably no arguments) given
  if ( missing(aggregate) ) return(NA)

  # choke if arguments out of range
  if ( pure < 1 || aggregate < 0 ) {
    message("  warn : xem.isTwoContained : either pure < 1 or aggregate < 0, values ", pure, " ", ↙
aggregate)
    warning("problematic arguments ", pure, " ", aggregate)
    return(FALSE)
  }

  # establish an upper limit power of two using the larger of 'pure' or 'aggregates'
  base <- 1
  max  <- max(aggregate, pure)
  while ( base < max ) base <- base * 2
  if ( DBUG > 2 ) message("base                : ", base)

  # fill vector using 'aggregate'
  reds <- integer()                           # vector
  lid  <- base                                # "lid"
  red  <- aggregate                           # scalar
  while ( lid >= 1 ) {
    if ( red >= lid ) {
      red <- red %% lid                       # modulus operator, remainder after division
      reds <- c(reds, lid)
    }
    else {
      reds <- c(reds, as.integer(0))
    }
    lid <- lid / 2                            # halving operation
    if ( DBUG > 2 ) message("lid just halved ", lid)
  }
  if ( DBUG > 1 ) { cat("reds (structure)    :"); str(reds) }

  # bail out when only 'aggregate' given
  if ( missing(pure) ) {
    return(reds)                              # CAUTION: 'pure' must remain unchanged to this point
  }

  # report
  if ( DBUG > 1 ) message("pure                : ", pure)
  if ( DBUG > 1 ) message("aggregate           : ", aggregate)

  # confirm pure
  ispure <- FALSE
  x      <- 1
  while ( x <= pure ) {
    if ( x == pure ) ispure <- TRUE
    x <- x * 2
  }

  # exit if not okay
  if ( ispure == FALSE ) {
    message("  warn : xem.isTwoContained : pure not pure ", pure)
    warning("pure not pure ", pure)
    return(FALSE)
  }
  else {
    if ( DBUG > 2 ) message("pure power of two ")
  }

  # hunt in vector
  hit <- match(pure, reds, nomatch = 0)
  ret <- as.logical(hit)
```

```
  if ( DBUG > 1 ) message("hit                 : ", hit)
  if ( DBUG > 1 ) message("ret                 : ", ret)

  # return
  ret
}

# ---------------------------------
#  function : tst.pyramidSort
# ---------------------------------

tst.pyramidSort <- function()
{
  numbers <- c(1, 3, 7, 8, 9, 12, 13)
  peak    <- 8
  output  <- xem.pyramidSort(numbers, peak)

  cat("numbers : ", numbers, "\n")
  cat("peak    : ", peak,    "\n")
  cat("output  : ", output,  "\n")
}

# ---------------------------------
#  function : tst.isTwoContained
# ---------------------------------

tst.isTwoContained <- function ()
{
  if ( DBUG > 1 ) cat("\n")
  xem.isTwoContained(13, 4)
  if ( DBUG > 1 ) cat("\n")
  xem.isTwoContained(11, 4)
  if ( DBUG > 1 ) cat("\n")
  xem.isTwoContained(0, 1)
  if ( DBUG > 1 ) cat("\n")
  xem.isTwoContained(123455, 2^16)           # 65536, TRUE
  if ( DBUG > 1 ) cat("\n")
  xem.isTwoContained(4.4, 4.4)
  if ( DBUG > 1 ) cat("\n")
  xem.isTwoContained(13, 5)
  if ( DBUG > 1 ) cat("\n")

  thirteen <- xem.isTwoContained(13)
  if ( DBUG > 1 ) cat("reduced vector      :", thirteen, "\n\n")

  zero <- xem.isTwoContained(0)
  if ( DBUG > 1 ) cat("reduced vector      :", zero, "\n\n")

  null <- xem.isTwoContained()
  if ( DBUG > 1 ) cat("reduced vector      :", null, "\n")
  if ( DBUG > 1 ) cat("\n")
}

# ---------------------------------
#  function : tst.utils
# ---------------------------------

tst.utils <- function (dbug)
{
  DBUG <<- dbug
  message("  * * * * * * * * * *\n  test : tst.utils : test commencing / DBUG = ", DBUG)

  str <- xem.trim("  abc def  ")
  xem.beep("alert")
  Sys.sleep(2)
  xem.beep("error")
  Sys.sleep(2)
  xem.beep("weird")
  xem.linesOfCode()
  xem.svnver()
  tst.pyramidSort()
  tst.isTwoContained()

  message("  test : tst.utils : test complete")
}

#  $Id: utils.R 5116 2010-08-18 09:45:29Z robbie $
#  end of file
```

```
#  file-purpose     : point of contact, user functions, call 'xeona' as required
#  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
#  file-create-date : Thu 04-Jun-2009 19:56 UTC
#  file-status      : working
#  file-keywords    : xeona R

#  $Revision: 5123 $
#  $Date: 2010-08-18 18:27:29 +0200 (Wed, 18 Aug 2010) $
#  $Author: robbie $
#  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/xeonar/xem.R $

#  PURPOSE
#
#  point of entry for client scripts
#
#  REQUIRES
#
#  * environment variable 'XEONAR' be appropriately set
#  * a standard R installation
#  * 'dot' (from 'Graphviz') if graph visualization is to be used
#  * 'display' (from 'ImageMagick') (until the code migrates to R)
#
#  TYPICAL USAGE
#
#  robbie.clean()  # CAUTION: must come first
#  source((file.path(Sys.getenv("XEONAR"), "xem.R"))
#  xem.job("model.xem", dbug = 2)
#
#  see also R script 'gaia.R' and similar
#
#  CLEAN-UP
#
#  no 'robbie.clean' calls are made in this code
#
#  UML DATA DIAGRAM
#
#  search on 'uml-dataset.png' and 'xeona-recset_00.xmi'
#
#  JOB CODES
#
#       0 = silent
#       1 = summarize
#       2 = plot all
#       4 = plot list
#       8 = breadplot
#      16 = graphviz diagram
#
#  TESTED USING
#
#  Initial development ('sojus')
#
#      R version 2.3.1 (2006-06-01)
#      Ubuntu 6.10 / Linux 2.6.17-12-generic
#      Graphviz version 2.22.2 (20090313.1817)
#      ImageMagick 6.2.4 10/02/07
#
#  Later development ('hinau')
#
#      note older graphviz package / see 'change-log.sojus' entry 2009-06-15
#
#      R version 2.10.1 (2009-12-14)
#      Ubuntu 10.04  Linux 2.6.32-24-generic (etc)
#      dot - Graphviz version 2.20.2 (Tue Mar  2 19:03:41 UTC 2010)
#      ImageMagick 6.5.7-8 2009-11-26 Q16

message("  file : xem.R : starting")

# ---------------------------------
#  variable : xem.colors
# ---------------------------------

# based on X11 and must be acceptable to both R and
# 'graphviz' (best to work from the 'graphviz' list)

xem.colors <- c(# default (1) - gray
                "dimgray",

                # authority (4) - reds
                "coral3",
                "tomato",
```

```
                "orange",
                "gold",

                # interface (3) - blues
                "mediumslateblue",
                "steelblue3",
                "cadetblue3",

                # context (2) - green
                "olivedrab3",
                "olivedrab4")

# ---------------------------------
#  function : xem.colors.print
# ---------------------------------

xem.colors.print <- function()
{
  x11()
  main <- paste("xem arrow color test")
  sub  <- "using the subset of x11 color names supported by graphviz"
  colorkey <<- sprintf("%2d \xb7 %s", seq(length(xem.colors)), xem.colors)
  pie(rep(1, length(xem.colors)), col = xem.colors, main = main, labels = colorkey)
  mtext(sub)
}

# ---------------------------------
#  function : xem.x11
# ---------------------------------
#  description  : control x11 plot frames (as a type of device)
#  role         : usually an early call for a particular job
#  status       : complete
# ---------------------------------

xem.x11 <- function (snooze)
{
  # slide show
  Sys.sleep(snooze)

  # host specific settings
  host <- Sys.info()["nodename"]
  if      ( host == "sojus" ) x11(width = 12, height = 8)
  else if ( host == "hinau" ) x11(width = 14, height = 7)
  else                        x11()

  # can add 'par' statements here
  par(col.main = "darkslategray")
}

# ---------------------------------
#  function : xem.saveplot
# ---------------------------------
#  description  : control x11 plot frames (as a type of device)
#  role         : usually an early call for a particular job
#  status       : complete
# ---------------------------------

xem.saveplot <- function (type = "notset",
                          dbug = 0)
{
  # source support file
  source(file.path(Sys.getenv("XEONAR"), "saveplots.R"), local = TRUE)     # save plot windows

  # report
  saveplot.report <- character()             # used for completion reporting

  # preparation
  filestub <- sub("\\.xem$", "", xemfile)    # trim trailing ".xem"

  # loop file formats
  if ( dev.cur() > 1 ) {                     # null device is number 1
    xem.saveplot.png(device = dev.cur(), filestub = filestub, type = type, dbug = dbug)
    xem.saveplot.pdf(device = dev.cur(), filestub = filestub, type = type, dbug = dbug)
    xem.saveplot.svg(device = dev.cur(), filestub = filestub, type = type, dbug = dbug)
  }

  # report
  message("  info : xem.saveplot : summary:")
  message(saveplot.report)
}
```

```
# --------------------------------
#  function : xem.job
# --------------------------------
#  description  : main function
#  role         : point of contact call
#  takes        : see signature
#  returns      : 'xmode'
#  techniques   : see code
#  status       : complete
# --------------------------------

xem.job <- function (xemfile,
                     xeona  = "xeona.mach",  # for default use, must be on the system 'PATH'
                     mlabel = "xem.R trial", # main label in plots
                     dbug   = 0,             # override default value set above
                     jobs   = NULL,          # override XEM file 'program.r-processing.r-policy'
                     snooze = 0,
                     report = 0)
{
  # --------------------------------
  #  preamble
  # --------------------------------

  # set debug level here
  DBUG <- dbug

  # report
  if ( is.null(jobs) ) jobrep <- "(XEM file)"
  else                 jobrep <- jobs
  message("  info : xem.job : commencing with",
          " xemfile = '" , xemfile, "'",
          " mlabel = "   , mlabel,
          " / DBUG = "   , DBUG,
          " / jobs = "   , jobrep,
          " / snooze = " , snooze,
          " / report = " , report)
  on.exit(message("  info : xem.job : complete (on.exit)"))
  message("  info : xem.job : nodename = " , Sys.info()["nodename"])

  # --------------------------------
  #  source
  # --------------------------------

  source(file.path(Sys.getenv("XEONAR"), "utils.R"),     local = TRUE)    # utility functions
  source(file.path(Sys.getenv("XEONAR"), "run.R"),       local = TRUE)    # run calls
  source(file.path(Sys.getenv("XEONAR"), "dataproc.R"),  local = TRUE)    # nested dataset
  source(file.path(Sys.getenv("XEONAR"), "extract.R"),   local = TRUE)    # extract data, normal plots
  source(file.path(Sys.getenv("XEONAR"), "breadplot.R"), local = TRUE)    # breadboard plot
  source(file.path(Sys.getenv("XEONAR"), "graphviz.R"),  local = TRUE)    # graphviz visualization

  # --------------------------------
  #  core establishment calls
  # --------------------------------

  xver   <<- xem.svnver()

  if ( is.null(xeona) ) {
    xemfile <<- xem.xemconf.model(xemfile)             # reports on but does no change 'xemfile' string
    xcall   <<- "na"                                   # normally set as side-effect of 'xem.xeona'
    xmode   <<- "na"                                   # normally set by 'xem.freshen'
  }
  else {
    xemfile <<- xem.xemconf.guard(xemfile)             # CAUTION: can change 'xemfile' string
    xmode   <<- xem.freshen(xeona, xemfile, report)    # generate a fresh deep or perhaps shallow XEM
file
  }

  xdata  <<- xem.slurp(xemfile)                        # slurp XEM file
  xset   <<- xem.loop(xdata)

  xsteps <<- xem.getValue(xset, "time-horizon", "steps")

  # --------------------------------
  #  recset reporting
  # --------------------------------

  message("recset length  : ", sprintf("%4d", length(xset)))
  message("data lines     : ", sprintf("%4d", length(xdata)))

  if ( DBUG > 1 ) {
    message()
```

```
  message("structure")
  message()
  str(xset)
  message()
}
if ( DBUG > 0 ) {
  message("summary")
  print(summary(xset))                      # CAUTION: 'print' wrapper needed
  message()
}

# ---------------------------------
#  job specification
# ---------------------------------

jobTerms <- c("summarize", "plot all", "plot list", "breadplot", "graphviz")

xpolicy <<- xem.getPolicy(xset)
xtitle  <<- xem.getValue(xset, "r-processing", "r-title")

if ( length(xpolicy) == 0 ) stop("no run policy found in xem file (malformed data)")
if ( is.null(jobs) ) jobs <- xpolicy       # function argument takes priority over XEM file value

jobsDecomp <- xem.isTwoContained(jobs)
jobsDecomp <- append(jobsDecomp, rep(0, 6 - length(jobsDecomp)), after = 0)   # pad to length 6
jobMsgs    <- paste(jobTerms[rev(jobsDecomp) > 0], collapse = " * ")
message("jobs           : ", jobs, " = ", jobsDecomp, " = ", jobMsgs)         # values like 008401

# ---------------------------------
#  job 1 : summarize
# ---------------------------------

if ( xem.isTwoContained(jobs, 1) ) {
  message("  info : xem.job : summarize job starts")
  xem.x11(snooze)
  xsummary <<- xem.summarize(xset)
  xem.saveplot("summary", DBUG)
}

# ---------------------------------
#  job 2 : plot all (but will truncate if necessary)
# ---------------------------------

if ( xem.isTwoContained(jobs, 2) ) {
  message("  info : xem.job : plot all job starts")
  xem.x11(snooze)
  xem.plotAllTs(xset, kind = "in")
  xem.saveplot("in", DBUG)
  xem.x11(snooze)
  xem.plotAllTs(xset, kind = "out")
  xem.saveplot("out", DBUG)
}

# ---------------------------------
#  job 4 : plot list
# ---------------------------------

# loop to plot list
if ( xem.isTwoContained(jobs, 4) ) {
  message("  info : xem.job : plot list job starts")

  # grab data
  plotStr <- xem.getPlotList(xset)
  if ( nchar(plotStr) > 0 ) {

    # split single strings
    sep     <- "[[:blank:]]+"               # define the split regex
    plots   <- unlist(strsplit(plotStr, split = sep))

    # loop
    for ( i in 1:length(plots) )
      {
        if ( nchar(plots[i] ) == 0 ) next    # protect against empty strings, namely the XEM value ""
        parts <- xem.splitFieldname(plots[i])
        xem.x11(snooze)
        xem.plotValue(xset, parts[2], parts[3])
        xem.saveplot("plotlist", DBUG)
      }
  }
  else {
    message("  warn : xem.job : plot list is empty, abandoning job")
```

```
    }
  }

  # ---------------------------------
  #  job 8 : breadplot
  # ---------------------------------

  if ( xem.isTwoContained(jobs, 8) ) {
    message("  info : xem.job : breadplot job starts")
    xem.x11(snooze)
    xcapture <<- xem.breadplot(xset, xtitle)
    xem.saveplot("breadplot", DBUG)
  }

  # ---------------------------------
  #  job 16 : graphviz
  # ---------------------------------

  if ( xem.isTwoContained(jobs, 16) ) {
    message("  info : xem.job : graphviz job starts")
    Sys.sleep(snooze)
    # xem.x11(snooze)
    # plot.new()
    xret <<- xem.graphviz(xset, xtitle)
  }

  # ---------------------------------
  #  completion reporting
  # ---------------------------------

  if ( TRUE ) {
    message("xeona call     : ", xcall)
    message("xem steps      : ", xsteps)

    message("debugging      : DBUG set to ", DBUG)

    message("call           : str(xset)")
    message("call           : summary(xset)")
  }

  # ---------------------------------
  #  housekeeping
  # ---------------------------------

  xem.beep("alert")

  # TOFIX: 17-Aug-2010: temporary settings

  Sys.sleep(1)                              # TOFIX: 15-Aug-2010: temporary
  system("killall display")                 # kill graphviz plot window

  invisible(xmode)
}

message("  file : xem.R : finishing")

#  $Id: xem.R 5123 2010-08-18 16:27:29Z robbie $
#  end of file
```

```
;;; xeona.el --- central point of contact for all xeona emacs support

;  file-purpose     : central point of contact for all xeona emacs support
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Thu 22-Jul-2010 07:37 UTC
;  file-status      : working
;  file-keywords    : emacs xeona

;  $Revision: 5180 $
;  $Date: 2010-08-24 14:46:56 +0200 (Tue, 24 Aug 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xeona.el $

;  XEONA CODEBASE COPY
;
;  * this code originated as part of the xeona codebase
;
;  * the GLPv2 license included here derives from Emacs Lisp
;    coding practice and from not the 'xeona' project --
;    users can optionally chose to adopt the 'xeona' norm
;    instead
;
;  * if this file is kept in your subversion working copy,
;    you may need to tell Emacs where to find it -- see
;    below for a representative '.emacs' entry

;; Copyright (C) 2010 Robbie Morrison

;; This file is *NOT* part of GNU Emacs.

;; This file is free software; you can redistribute it
;; and/or modify it under the terms of the GNU General
;; Public License as published by the Free Software
;; Foundation; either version 2, or (at your option)
;; any later version.

;; This file is distributed in the hope that it will be
;; useful, but WITHOUT ANY WARRANTY; without even the
;; implied warranty of MERCHANTABILITY or FITNESS FOR A
;; PARTICULAR PURPOSE.  See the GNU General Public
;; License for more details.

;; You may have received a copy of the GNU General
;; Public License along with GNU Emacs; see the file
;; COPYING.  If not, write to the Free Software
;; Foundation, Inc., 59 Temple Place - Suite 330,
;; Boston, MA 02111-1307, USA.

;;; Commentary:

;;  ToDos
;;
;;      * complete the decoupling from my '~/.emacs' file
;;        call : $ emacs --no-init-file \
;;                       --load /path/to/xeona.el \
;;                       --file submodel.15.guard.xem
;;        then : M-x xem-mode
;;               M-x xog-mode
;;
;;      * run `checkdoc' periodically
;;
;;  Known issues
;;
;;  Emacs configuration file (.emacs) entry
;;
;;    (defconst xeona-elisp-directory
;;      (substitute-in-file-name
;;       "$HOME/$SYNK/xeona/svn2/futz/trunk/elisp/xeona.el"))
;;
;;    (if (file-readable-p xeona-elisp-directory)
;;        (load-file xeona-elisp-directory)
;;      (message "  robbie: WARNING: cannot load %s" xeona-elisp-directory))

;;; History:

;;      * see the subversion log messages

;;; Code:

;; ---------------------------------
;;  file associations
```

```
;; ---------------------------------

;; the \' matches the end of a string, where as $ matches the
;; empty string before a newline -- thus $ may lead to unexpected
;; behavior when dealing with filenames containing newlines

(setq auto-mode-alist (cons '("\\.xem\\'" . xem-mode) auto-mode-alist))
(setq auto-mode-alist (cons '("\\.xog\\'" . xog-mode) auto-mode-alist))

;; ------------------------------------------------------------------------
;;  overarching support for Xem and Xog major modes
;; ------------------------------------------------------------------------

;; list of files to load -- the 'dev-tests' are strictly for testing and can be omitted

(defconst xeona-file-stubs
  '("xem" "xem-1" "xem-2" "xem-3" "xog" "xrstat" "xumber" "dev-tests")
  "List of file stubs to load on startup.")

;; our common location

(defconst xeona-file-stem
  (file-name-directory xeona-elisp-directory)               ; how we got here!
  "Current directory.")

;; load everything

(defun xeona-load-files (filestubs)
  "Load all FILESTUBS."
  (let* ((stub   nil )
         (loops  0   ))
    (dolist (stub filestubs)
      (incf loops)
      (setq file (concat xeona-file-stem stub ".el"))  ; the ".el" is required
      (load-file file))                                ; this call reports to '*Messages*'
    (message "  xeona: load-files: should have just loaded %d files" loops) ))

;; actual call

(xeona-load-files xeona-file-stubs)

;; ------------------------------------------------------------------------
;;  overarching support for Xem and Xog major modes
;; ------------------------------------------------------------------------

;; the two hook functions can be moved back to the user's
;; configuration file and fleshed out if need be

;; Xem mode

(add-hook 'xem-mode-hook 'xem-mode-customization)

(defun xem-mode-customization ()
  "Xem mode customization."
  (message "  robbie: xem-mode-customization: complete (also hollow)"))

;; Xog mode

(add-hook 'xog-mode-hook 'xog-mode-customization)

(defun xog-mode-customization ()
  "Xog mode customization."
  (message "  robbie: xog-mode-customization: complete (also hollow)"))

;; ------------------------------------------------------------------------
;;  save non-visited buffers
;; ------------------------------------------------------------------------

(defconst xem-buffers-visit-alist
  '((xem-buffer-xeona-output     . ".xog")
    (xem-buffer-r-capture        . ".rog")
    (xem-buffer-dump-tar-capture . ".tog"))
  "Alist of buffer names and file extensions.")

(defun xem-buffers-visit-all ()
  "Save any buffers defined in `xem-buffers-save-all' which are currently open.

This call also destroys the original buffer names in the process."
  (interactive)
  (let* ((filename  (buffer-file-name nil)              )   ; current buffer
         (filestub  (file-name-sans-extension filename) )
```

```
         (buffers   xem-buffers-visit-alist             )
         (buffname  nil                                 )
         (savename  nil                                 )
         (loops     0                                   )
         (plural    "s"                                 ))
    (dolist (buffer buffers)
      (setq buffname (symbol-value (car buffer))) ; CAUTION: note function 'symbol-value'
      (setq savename (concat filestub (cdr buffer)))
      (setq bf (get-buffer buffname))        ; non-nil if 'buffname' exists
      (when bf
        (incf loops)
        (set-buffer bf)                      ; make buffer current for editing
        (write-file savename)
        (rename-buffer buffname nil)         ; require uniqueness TOFIX: 19-Aug-2010: confirm action
        (message "  xem: buffers-visit-all: saved buffer '%s' as '%s'" buffname savename)))
    (if (= loops 1) (setq plural ""))
    (message "  xem: buffers-visit-all: complete with %d save%s" loops plural) ))

(defun xeona-list-buffers ()
  "Test function to slowly list 'xeona-buffers-regex'."
  (interactive)
  (let* ((regexs  xeona-buffers-regexs )
         (loops   0                    )
         (plural  nil                  ))
    (dolist (regex regexs)
      (incf loops)
      (message "  xeona: list-buffers: buffer %02d regex '%s'" loops regex)
      (sit-for 2))                           ; wait some
    (if (= loops 1) (setq plural "") (setq plural "s"))
    (message "  xeona: list-buffers: complete with %d showing%s." loops plural)
    loops ))                                 ; expose potential return value

;; ---------------------------------------------------------------------------
;;  across-mode buffer cycling
;; ---------------------------------------------------------------------------

;; CAUTION: setting global key binds is not neighborly but it
;; does save providing a special minor mode for each buffer -
;; that said, this really should be revisited

(global-set-key [(        f3)] 'xeona-cycle-buffers)

;; CAUTION: function 'xeona-regex-buffername' must be
;; parsed ahead of constant 'xeona-buffers-regexs'

(defun xeona-regex-buffername (buffername)
  "Function to regexize a BUFFERNAME.

That is replace all \"*\" with \"\\*\"."
  (let* ((regex  (replace-regexp-in-string "\\*" "\\*" buffername nil t) ))     ; [1]
    (message "  xeona: regex-buffername: '%s' to '%s'." buffername regex)
    regex ))                                 ; CAUTION: expose return value
    ;; [1] CAUTION: t for 'fixedcase' in 'rep' is essential

;; the buffers 'xumber-sum-buffer' and 'xumber-history-buffer'
;; are not accessible from here but are instead duplicate
;; hardcoded -- upstream changes will not cause problems because these

(defconst xeona-buffers-regexs
  (list "\\.xem$"
        "\\.xog$"
        "\\.rog$"
        (xeona-regex-buffername xem-buffer-xeona-output)
        (xeona-regex-buffername xem-buffer-xeona-data-rules)
        (xeona-regex-buffername xem-buffer-xeona-usage)
        (xeona-regex-buffername xem-buffer-xem-diff)
        (xeona-regex-buffername xem-buffer-r-capture)
        (xeona-regex-buffername xem-buffer-dump-tar-capture)
        (xeona-regex-buffername xem-buffer-dump-review)
        (xeona-regex-buffername "*xumber-tally*")
        (xeona-regex-buffername "*xumber-history*")
        (xeona-regex-buffername "*Shell Command Output*")   ; optional
        (xeona-regex-buffername "*Async Shell Command*")    ; optional
        (xeona-regex-buffername "*Process List*")           ; optional
        (xeona-regex-buffername "*Messages*")               ; optional
;;;     "\\*xumber-.*\\*"
        ) "Potentially open xeona buffers.")

(defun xeona-cycle-buffers ()
  "Cycle to next appropriate buffer and then maximize it.
```

```
This function also applies a darker background to guard models."
  (interactive)
  (let* ((guard-bgcolor  "beige"              )   ; subtle
         (guard-bgcolor  "sienna"             )   ; provides good text contrast
         (regexs         xeona-buffers-regexs )
         (rsize          (length regexs)      )
         (blist          (buffer-list)        )
         (bsize          (length blist)       )
         (bc             (current-buffer)     )   ; lisp buffer object
         (bf             nil                  )   ; lisp buffer object
         (regex          nil                  )
         (hit            nil                  ))  ; to pass 'string-match' return upstream
    (message "  xeona: cycle-buffers: regexs %d, buffers %d" rsize bsize)
    (pop blist)                                  ; discard the current buffer
    (while (and (setq bf (pop blist))            ; processed in order
                (dolist (regex regexs (null hit))
                  (if (string-match regex (buffer-name bf)) (setq hit t)))))
    (if bf                                       ; CAUTION: must protect against nil buffer
        (progn
          (bury-buffer bc)                       ; OPTIONAL: active = cycle, disable = toggle
          (switch-to-buffer bf)                  ; switch
          (delete-other-windows)                 ; make full-size
          ;; coloration conditional for guard files
          (if (string-match "\\.guard\\.xem$" (buffer-name bf))
              (set-background-color guard-bgcolor)
            (set-background-color "white"))
          (message nil)                          ; hack to resize minibuffer to one line
          (message "  xeona: cycle-buffers: next buffer found '%s'." (buffer-name bf)))
      (message "  xeona: cycle-buffers: no next buffer found.")) ))

;; CAUTION: you can only apply a background color to a frame in
;; emacs, you cannot do this at the granularity of windows or
;; buffers -- hence you need to cycle out of guard file buffers
;; using the 'xeona-cycle-buffers' command if you want your
;; standard background color to return

;;; xeona.el ends here

;  $Id: xeona.el 5180 2010-08-24 12:46:56Z robbie $
;  end of file
```

```
;;; xem.el --- editing support for xeona model files

;  file-purpose     : emacs editing support for xeona model files
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Fri 16-Jul-2010 15:19 UTC
;  file-status      : ongoing
;  file-keywords    : emacs xeona

;  $Revision: 5309 $
;  $Date: 2010-10-13 10:03:46 +0200 (Wed, 13 Oct 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xem.el $

;  XEONA CODEBASE COPY
;
;  * this code was originally imported from RCS 1.48 of
;    '/home/robbie/synk/xeona/xem-processing/xem.el'
;
;  * the GLPv2 license included here derives from Emacs Lisp
;    coding practice and from not the 'xeona' project --
;    users can optionally chose to adopt the 'xeona' norm
;    instead

;; Copyright (C) 2009-2010 Robbie Morrison

;; This file is *NOT* part of GNU Emacs.

;; This file is free software; you can redistribute it
;; and/or modify it under the terms of the GNU General
;; Public License as published by the Free Software
;; Foundation; either version 2, or (at your option)
;; any later version.

;; This file is distributed in the hope that it will be
;; useful, but WITHOUT ANY WARRANTY; without even the
;; implied warranty of MERCHANTABILITY or FITNESS FOR A
;; PARTICULAR PURPOSE.  See the GNU General Public
;; License for more details.

;; You may have received a copy of the GNU General
;; Public License along with GNU Emacs; see the file
;; COPYING.  If not, write to the Free Software
;; Foundation, Inc., 59 Temple Place - Suite 330,
;; Boston, MA 02111-1307, USA.

;;; Commentary:

;;  ToDos
;;
;;      * check current buffer logic (although no problems have
;;        been detected), while noting the warning in section 27.2
;;        of the elisp manual
;;
;;      * remove redundant `message' development calls
;;
;;      * add `abbreviate-file-name' to messages where appropriate
;;
;;      * run `checkdoc' periodically
;;
;;  Known issues
;;
;;      * IGNORE: xeona reorders its XEM file on first writing, which means
;;        that the 'vhold' and 'hhold' placeholders becomes scrambled -- any
;;        fix would require an semantic understanding of location and that is
;;        not a priority at this stage of development
;;
;;  Major mode programming resources
;;
;;      * Cameron et al (2004 p381+) for an introduction
;;      * "info elisp" and in particular "Major Mode Examples"
;;      * Xah's website: http://xahlee.org/emacs
;;      * Emacs wiki: www.emacswiki.org/emacs/MajorMode
;;
;;  Note on `message' calls
;;
;;      Any `message' call expected to be read (meaning, should
;;      remain in the mini-buffer long enough to be seen), should
;;      end with a '.' dot.
;;
;;  References
;;
```

```
;;     Cameron, Debra, James Elliott, Marc Loy, Eric Raymond, and Bill
;;         Rosenblatt.  2005.  Learning GNU Emacs -- Third edition.
;;         O'Reilly and Associates, Sebastopol, CA, USA.  ISBN:
;;         0-596-00648-9.  [For GNU Emacs 21.3.5]

;;; History:

;;      * see the subversion log messages

;;; Code:

;; ---------------------------------
;;  user-modifiable global constants
;; ---------------------------------

(defconst xem-website
  "http://www.iet.tu-berlin.de/deeco"        ; currently set to 'deeco'!
  "Xeona website URL.")

(defconst xem-dchar
  "#"
  "Xeona disable char (as set in 'common.cc').")

(defconst xem-model-extension
  ".xem"
  "Xeona model extension (as set in 'common.cc').")

(defconst xem-backup-tag
  "~"
  "Xeona backup tag (as set in 'common.cc').")

(defconst xem-model-guard-tag
  ".guard"
  "Xeona guard file tag (as set in 'common.cc').")

(defconst xem-binary-name
  "xeona.mach"
  "Xeona binary name (located semi-intelligently).")

(defconst xem-buffer-xeona-output
  "*xeona-output*"
  "Buffer for xeona output.")

(defconst xem-buffer-xeona-data-rules
  "*xeona-data-rules*"
  "Buffer for xeona data rules.")

(defconst xem-buffer-xeona-usage
  "*xeona-usage*"
  "Buffer for xeona usage.")

(defconst xem-buffer-xem-diff
  "*xem-diff*"
  "Buffer for xem diff results.")

(defconst xem-file-stem
  xeona-file-stem
  "Current directory.")

(defconst xem-xumber-file
  (concat xem-file-stem "xumber.el")
  "Xumber mode file to drag in.")

;; ---------------------------------
;;  requirements
;; ---------------------------------

(require 'ediff)                           ; "a comprehensive visual interface to diff & patch"
(require 'hi-lock)                         ; "minor mode for interactive automatic highlighting"
(require 'thingatpt)                       ; "get the `thing' at point"
(require 'highlight-current-line)          ; "highlight line where the cursor is"

;; ---------------------------------
;;  mode function
;; ---------------------------------

(defun xem-mode ()
  "Major mode for editing xeona XEM model files.

All interactive functions start \"xem-\" for easy identification.
```

```
If the menu bar is hidden, a background Xem menu can be brought
up with: \"Ctrl-Shift-leftmouse\".

Dedicated key binding are indicated in the Xem menu.  For a
complete listing, use \\[describe-bindings].

The cursor changes from black to turquoise under the selective
display.  The cursor changes from a box to a line under narrowing.

Much of the functionality of this mode depends on access to a
suitable 'xeona' binary.  This access is tested on loading and
any failure in noted.

The 'xem-toggle-narrow' function requires 'narrow-to-region' to
be enabled.  In which case you may need to the following in your
emacs configuration file:

  (put 'narrow-to-region 'disabled nil)"
  (interactive)

  (message "  xem: xem-mode setup commencing")

  ;; recommended
  (kill-all-local-variables)

  ;; information
  (message "  xem: default-face-height %d" xem-default-face-height)

  ;; emacs stuff
  (setq major-mode 'xem-mode)                ; used by C-h m aka 'describe-mode'
  (setq mode-name  "Xem")
  (use-local-map    xem-mode-map)
  (run-hooks       'xem-mode-hook)           ; optionally defined in the user's configuration file

  ;; use our custom menu, "Xem", located to the right of "Tools"
  (menu-bar-mode 1)

  ;; visual bell (particularly as the Ubuntu system bell is broken as of mid-2010)
  (setq visible-bell t)

  ;; background (can be redefined within 'xem-mode-hook')
  (setq fill-column 63)                      ; auto-fill wrap (should strictly speaking be 65 - 4)
  (setq selective-display-ellipses nil)      ; omit "..."

  ;; global variables
  (setq xem-binary   (xem-locate-binary xem-binary-name))
  (setq xem-svn      (xem-binary-svn xem-binary))
  (setq xem-have-run (xem-have-i-run))

  ;; make your comment command use the same shortcut for 'comment-dwim' (dwim = do what I mean),
  ;; the user may have changed their default
  ;; (define-key xem-mode-map [remap comment-dwim] 'xem-comment-dwim)

  ;; miscellaneous
  (setq message-log-max t)                   ; 'true' means do not truncate "*Messages*"
  (global-auto-revert-mode 1)                ; experimental, note also 'auto-revert-mode'
  (message "  xem: xem-mode: Global-Auto-Revert mode activated")
  (xem-toggle-SITFOR +1)

  ;; general actions
  (xem-colorize)                             ; automatically colorize buffer on opening
; (xem-maxscreen-me)
; (xem-fullscreen-me)

  ;; if-run actions
  (if (xem-have-i-run)                       ; 'true' if XEM file has run
      (message "  xem: xem-mode: XEM file deemed to have run")
    (message "  xem: xem-mode: XEM file deemed NOT to have run"))

  ;; Xumber mode
  ; both work the same
  (autoload 'xumber-mode
    xem-xumber-file
    "Toggle xeona number mode."
    t)                                       ; 't' for interactive
  (xumber-mode t)
  (if xumber-mode
      (message "  xem: xem-mode: Xumber mode now active")
    (message "  xem: xem-mode: Xumber mode INACTIVE"))

  ;; completion reporting
```

```
  (if (not xem-binary) (ding))               ; acknowledge visual bell setting
  (message "  xem: xem-mode setup complete (binary %s revision %s)." xem-binary xem-svn))

;; ---------------------------------
;;  custom key maps
;; ---------------------------------

(defvar xem-mode-map nil "Local keymap for Xem mode buffers.")
;; CAUTION: must NOT be (make-local-variable 'xem-mode-map)

;; see additional file 'keys-and-keymaps.txt' for more
;; information on this topic

(if xem-mode-map
    ()                                       ; protection against double loading

  ;; keyboard shortcuts 1
  (setq xem-mode-map (make-sparse-keymap))
  (define-key xem-mode-map [(shift          next)] 'xem-move-entity-next)
  (define-key xem-mode-map [(shift         prior)] 'xem-move-entity-prior)
  (define-key xem-mode-map [(shift           tab)] 'xem-toggle-data)
  (define-key xem-mode-map [(shift           end)] 'xem-move-hash-ring)
  (define-key xem-mode-map [(shift          home)] 'xem-cycle-squish)
  (define-key xem-mode-map [(shift         right)] 'xem-move-hash-next)
  (define-key xem-mode-map [(shift          left)] 'xem-move-hash-prior)
  (define-key xem-mode-map [?\S-                ] 'xem-toggle-rules)

  ;; keyboard shortcuts 2
  (setq xem-mode-map (make-sparse-keymap))   ; CAUTION: resets the above
  (define-key xem-mode-map [(shift         right)] 'xem-cycle-squish)
  (define-key xem-mode-map [(shift          left)] 'xem-toggle-rules)
  (define-key xem-mode-map [(            backtab)] 'xem-toggle-data)            ; type (shift tab)
  (define-key xem-mode-map [(shift            up)] 'xem-move-entity-prior)
  (define-key xem-mode-map [(shift          down)] 'xem-move-entity-next)
  (define-key xem-mode-map [(shift          home)] 'xem-move-hash-prior)
  (define-key xem-mode-map [(shift           end)] 'xem-move-hash-next)
  (define-key xem-mode-map [(shift         prior)] 'xem-move-field-in-prior)
  (define-key xem-mode-map [(shift          next)] 'xem-move-field-in-next)
  (define-key xem-mode-map [(shift         pause)] 'xem-toggle-highlight-output)
  (define-key xem-mode-map [(control shift   end)] 'xem-move-hash-ring)
  (define-key xem-mode-map [(shift         print)] 'xem-toggle-narrow)

  (define-key xem-mode-map [(               menu)] 'xem-run-me)
  (define-key xem-mode-map [(shift          menu)] 'xem-r-run)
  (define-key xem-mode-map [(control        menu)] 'xem-dump)
  (define-key xem-mode-map [(control shift  menu)] 'xem-diff-me-ediff)

  ;; temporary key maps for convenience during development

  (define-key xem-mode-map [(                f2)] 'xem-reset-me)

  ;; custom menu (does not make use of the 'easymenu' elisp package)
  (defvar menuXem00 (make-sparse-keymap "Xem Menu"))    ; mouse label
  (defvar menuXem01 (make-sparse-keymap "Xem 01"))
  (defvar menuXem02 (make-sparse-keymap "Xem 02"))
  (defvar menuXem03 (make-sparse-keymap "Xem 03"))
  (defvar menuXem04 (make-sparse-keymap "Xem 04"))
  (defvar menuXem05 (make-sparse-keymap "Xem 05"))
  (defvar menuXem06 (make-sparse-keymap "Xem 06"))
  (defvar menuXem07 (make-sparse-keymap "Xem 07"))
  (defvar menuXem08 (make-sparse-keymap "Xem 08"))
  (defvar menuXem09 (make-sparse-keymap "Xem 09"))
  (defvar menuXem10 (make-sparse-keymap "Xem 10"))
  (defvar menuXem11 (make-sparse-keymap "Xem 11"))

  ;; mouse usage
  (define-key xem-mode-map [C-S-down-mouse-1] menuXem00)

  ;; top-level entry (visible in menu bar, right of "Tools")
  (define-key xem-mode-map [menu-bar xem]          (cons "Xem"                menuXem00)) ; menu-bar ⬀
label

  ;; main menu bottom (reverse order)
  (define-key menuXem00 [about]       '(menu-item "About"                   xem-menu-about           ⬀
:help "One-line report on Xem mode"))
  (define-key menuXem00 [help]        '(menu-item "Describe mode"           xem-menu-describe        ⬀
:help "Standard emacs 'describe-mode' call"))
  (define-key menuXem00 [bindings]    '(menu-item "Describe key bindings *" describe-bindings        ⬀
:help "Standard emacs 'describe-bindings' call"))
  (define-key menuXem00 [usage-filt]  '(menu-item "Show xeona usage"        xem-usage-filtered       ⬀
:help "Show regex filtered xeona usage using a call to \"xeona --usage\""))
```

```
  (define-key menuXem00 [help-data]   '(menu-item "Show xeona data rules"   xem-menu-show-data-rules ⬀
:help "Show xeona data rules using a call to \"xeona --data\""))
  (define-key menuXem00 [separator-1] '("--"))

  ;; sub-menu entries (reverse order)
  (define-key xem-mode-map [menu-bar xem submenu11] (cons "Miscellaneous"                  menuXem11))
  (define-key xem-mode-map [menu-bar xem submenu10] (cons "File"                           menuXem10))
  (define-key xem-mode-map [menu-bar xem submenu09] (cons "Model dumps"                    menuXem09))
  (define-key xem-mode-map [menu-bar xem submenu08] (cons "Visual post-processing using R" menuXem08))
  (define-key xem-mode-map [menu-bar xem submenu07] (cons "Run / diff / reset model"       menuXem07))
  (define-key xem-mode-map [menu-bar xem submenu06] (cons "Hi-light"                       menuXem06))
  (define-key xem-mode-map [menu-bar xem submenu05] (cons "Tidy code"                      menuXem05))
  (define-key xem-mode-map [menu-bar xem submenu04] (cons "Report environment"             menuXem04))
  (define-key xem-mode-map [menu-bar xem submenu03] (cons "Summarize model"                menuXem03))
  (define-key xem-mode-map [menu-bar xem submenu02] (cons "Insert"                         menuXem02))
  (define-key xem-mode-map [menu-bar xem submenu01] (cons "Navigate"                       menuXem01))

  ;; main menu top (reverse order)
  (define-key menuXem00 [separator-2] '("--"))
  (define-key menuXem00 [tog-screen]  '(menu-item "Full-screen (toggle)"        ⬀
xem-fullscreen-me-toggle    :help "Toggle full screen (Linux only)"))
  (define-key menuXem00 [tog-max]     '(menu-item "Maximize screen (toggle)"    ⬀
xem-maxscreen-me-toggle     :help "Toggle maximize screen (Linux only)"))
; (define-key menuXem00 [switch]      '(menu-item "Switch to useful buffer"     ⬀
xem-switch-to-useful-buffer :help "Switch to useful buffer OR show buffer list"))
  (define-key menuXem00 [cycle-bufs]  '(menu-item "Cycle to next useful buffer" xeona-cycle-buffers    ⬀
:help "Cycle to next useful buffer"))
  (define-key menuXem00 [narrow]      '(menu-item "Narrow to entity (toggle)"   xem-toggle-narrow      ⬀
:help "Toggle narrow to entity function (a line cursor indicates this action) (requires ⬀
'narrow-to-region' be enabled)"))
  (define-key menuXem00 [enable]      '(menu-item "Enabled status (toggle)"     xem-toggle-data        ⬀
:help "Toggle entity enabled status OR create duplicate disabled field or reinstate current disabled ⬀
field (access via <S-tab>)"))
  (define-key menuXem00 [toggle-rule] '(menu-item "Rule indent (toggle)"        xem-toggle-rules       ⬀
:help "Toggle rule indent between 1 and 2 cols (in order to change visibility under full collapse)"))
  (define-key menuXem00 [cycl-squish] '(menu-item "Cycle display collapse"      xem-cycle-squish       ⬀
:help "Cycle thru 3-way display collapse (a turquoise cursor indicates an incomplete display)"))

  ;; ---

  ;; submenu 1 (reverse order)
  (define-key menuXem01 [hash-cycle]  '(menu-item "Cycle thru disabled entities and fields" ⬀
xem-move-hash-ring       :help "Cycle to next disabled entity or field"))
  (define-key menuXem01 [hash-prior]  '(menu-item "Previous disabled entity or field"       ⬀
xem-move-hash-prior      :help "Move up to previous disabled entity or field"))
  (define-key menuXem01 [hash-next]   '(menu-item "Next disabled disable entity or field"   ⬀
xem-move-hash-next       :help "Move down to next disabled entity or field"))
  (define-key menuXem01 [outf-prior]  '(menu-item "Previous out-data field"                 ⬀
xem-move-field-out-prior :help "Move up to previous output data field"))
  (define-key menuXem01 [outf-next]   '(menu-item "Next out-data field"                     ⬀
xem-move-field-out-next  :help "Move down to next output data field"))
  (define-key menuXem01 [inf-prior]   '(menu-item "Previous in-data field"                  ⬀
xem-move-field-in-prior  :help "Move up to previous input data field"))
  (define-key menuXem01 [inf-next]    '(menu-item "Next in-data field"                      ⬀
xem-move-field-in-next   :help "Move down to next input data field"))
  (define-key menuXem01 [ent-prior]   '(menu-item "Previous entity"                         ⬀
xem-move-entity-prior    :help "Move up to previous entity"))
  (define-key menuXem01 [ent-next]    '(menu-item "Next entity"                             ⬀
xem-move-entity-next     :help "Move down to next entity"))

  ;; submenu 2 (reverse order)
  (define-key menuXem02 [uncamel]     '(menu-item "Uncamel current symbol"        xem-uncamel          ⬀
:help "Uncamel current symbol (meaning convert \"myId\" to \"my-id\")"))
  (define-key menuXem02 [pink]        '(menu-item "Colorized comment"             ⬀
xem-insert-strong-comment :help "Insert pink colorized comment"))
  (define-key menuXem02 [rule-scroll] '(menu-item "Rule via scroll"               ⬀
xem-insert-rule-scroll    :help "Insert rule by scrolling"))
  (define-key menuXem02 [rule-prompt] '(menu-item "Rule via prompt"               ⬀
xem-insert-rule-scroll    :help "Insert rule after prompting"))
  (define-key menuXem02 [call-scroll] '(menu-item "Entity via scroll"             ⬀
xem-insert-entity-scroll  :help "Insert new entity by scrolling an alphabetical list"))
  (define-key menuXem02 [call-regex]  '(menu-item "Entity via regex"              ⬀
xem-insert-entity-regex   :help "Insert new entity based on prompted case-sensitive regex"))
  (define-key menuXem02 [new-model]   '(menu-item "New model"                     xem-insert-new-model ⬀
:help "Insert new model (an empty buffer is recommended)"))

  ;; submenu 3 (reverse order)
  (define-key menuXem03 [scan-region] '(menu-item "List classes and remarks in region" ⬀
xem-scan-class-region       :help "Collate and list selected classes and builtin remarks"))
  (define-key menuXem03 [scan-buffer] '(menu-item "List classes and remarks in buffer" ⬀
xem-scan-class-buffer       :help "Collate and list all classes and builtin remarks"))
```

```
  (define-key menuXem03 [separator-3] '("--"))
  (define-key menuXem03 [identities]  '(menu-item "List field value identities"        ⬋
xem-occur-identities        :help "List identities used as field values (\"entity.\" is absent and ⬋
sub-entity identifiers may follow)"))
  (define-key menuXem03 [hashes]      '(menu-item "List hashes"                        ⬋
xem-occur-hashes            :help "List hashes"))
  (define-key menuXem03 [rules]       '(menu-item "List rules"                         xem-occur-rules ⬋
:help "List rules"))
  (define-key menuXem03 [classes]     '(menu-item "List classes"                       ⬋
xem-occur-classes           :help "List classes"))
  (define-key menuXem03 [disabled]    '(menu-item "List disabled entities"             ⬋
xem-occur-entities-disabled :help "List only disabled entities"))
  (define-key menuXem03 [entities]    '(menu-item "List entities"                      ⬋
xem-occur-entities          :help "List all entities"))
  (define-key menuXem03 [remarks]     '(menu-item "List builtin remarks"               ⬋
xem-occur-builtin-remarks   :help "List builtin remarks"))
  (define-key menuXem03 [separator-4] '("--"))
  (define-key menuXem03 [toggle-out]  '(menu-item "Show output highlights"             ⬋
xem-toggle-highlight-output :help "Toggle show output highlights as defined under the in-data field ⬋
'program.r-processing.r-highlight-output'"))
  (define-key menuXem03 [status]      '(menu-item "Report on run status"               ⬋
xem-menu-model-status       :help "Report on full file name and model run status"))

  ;; submenu 4 (reverse order)
  (define-key menuXem04 [about]       '(menu-item "Report on Xem mode"           xem-menu-about        ⬋
:help "Report on current Xem mode version number"))
  (define-key menuXem04 [svn-version] '(menu-item "Display xeona codebase"       xem-menu-svnversion   ⬋
:help "Display current xeona codebase svn version string"))
  (define-key menuXem04 [locate]      '(menu-item "Display current xeona binary" ⬋
xem-menu-report-binary :help "Display xeona binary in use by Xem mode"))

  ;; submenu 5 (reverse order)
  (define-key menuXem05 [clean]       '(menu-item "Clean model (obsolete)" xem-clean-model   :help ⬋
"Clean any \"(some problem)\" in the current model"))
  (define-key menuXem05 [revisit]     '(menu-item "Save and revisit buffer (to reset emacs local ⬋
variables)" xem-revisit         :help "Save and revisit the current buffer to reset the emacs local ⬋
variables"))
  (define-key menuXem05 [uncolorize]  '(menu-item "Uncolorize buffer *"                                ⬋
hi-lock-mode        :help "Remove color emphasis from buffer (by running 'hi-lock-mode')"))
  (define-key menuXem05 [colorize]    '(menu-item "Colorize buffer"                                    ⬋
xem-colorize        :help "Add color emphasis to buffer"))
  (define-key menuXem05 [reset-tabs]  '(menu-item "Reset tabstops, then save and revisit"              ⬋
xem-reset-tabstops  :help "Reset emacs local variables tabstop list using the prevailing format"))
  (define-key menuXem05 [reindent]    '(menu-item "Reindent buffer using tabstops"                     ⬋
xem-reindent        :help "Reindent entire buffer using emacs local variables tabstop list (save and ⬋
revisit file first if the tabstops have changed)"))
  (define-key menuXem05 [trim]        '(menu-item "Delete trailing whitespace"                         ⬋
xem-trim-trailing   :help "Delete trailing white space"))
  (define-key menuXem05 [squeeze]     '(menu-item "Squeeze blank lines"                                ⬋
xem-squeeze-lines   :help "Remove multiple blank lines without prompting"))

  ;; submenu 6 (reverse order)
  (define-key menuXem06 [kil-buffer]  '(menu-item "Clear buffer"    xem-hilighter-clear-buffer :help ⬋
"Clear all highlights"))
  (define-key menuXem06 [kil-region]  '(menu-item "Clear region"    xem-hilighter-clear-region :help ⬋
"Clear intersecting highlights from current region"))
  (define-key menuXem06 [hil-region]  '(menu-item "Hi-light region" xem-hilighter-region       :help ⬋
"Hilight current region"))
  (define-key menuXem06 [hil-entity]  '(menu-item "Hi-light entity" xem-hilighter-entity       :help ⬋
"Hilight current entity (except last) (becomes temporarily scrambled under display collapse)"))

  ;; submenu 7 (reverse order)
  (define-key menuXem07 [rebin]       '(menu-item "Change xeona binary"                          ⬋
xem-rebinary       :help "Change name of called binary"))
  (define-key menuXem07 [tidy]        '(menu-item "Tidy model using xeona"                       ⬋
xem-run-tidy       :help "Run current model using \"xeona --tidy\""))
  (define-key menuXem07 [reset-me]    '(menu-item "Reset model (do both the above)"              ⬋
xem-reset-me       :help "Intelligently zero and double-dot over entire buffer -- indicated whenever ⬋
'time-horizon.steps' is altered"))
  (define-key menuXem07 [zero-me]     '(menu-item "Zero out-data"                                ⬋
xem-zero-me        :help "Intelligently zero out-data over entire buffer"))
  (define-key menuXem07 [double-one]  '(menu-item "Truncate timeseries in-data on current line" ⬋
xem-double-dot-one :help "Truncate and add trailing double-dots to in-data on current line"))
  (define-key menuXem07 [double-dot]  '(menu-item "Double dot all timeseries in-data"           ⬋
xem-double-dot-me  :help "Intelligently add trailing double-dots to all in-data timeseries over entire ⬋
buffer"))
  (define-key menuXem07 [diff-me]     '(menu-item "Diff model (choose)"                          ⬋
xem-diff-me        :help "Prompt for diff method and then diff current model against its ~ file (user ⬋
is responsible for ensuring currency)"))
  (define-key menuXem07 [ediff-me]    '(menu-item "Ediff model"                                  ⬋
xem-diff-me-ediff  :help "Ediff current model against its ~ file (user is responsible for ensuring ⬋
```

```
currency)"))
  (define-key menuXem07 [run-1]       '(menu-item "Run model"                                    ⬋
xem-run-me         :help "Run current model after prompted for values for report|mode|yeek"))

  ;; submenu 8 (reverse order)
  (define-key menuXem08 [r-svgs]      '(menu-item "Open associated SVGs"              xem-r-svgs :help ⬋
"Open any associated SVG files"))
  (define-key menuXem08 [r-pdfs]      '(menu-item "Open associated PDFs"              xem-r-pdfs :help ⬋
"Open any associated PDF files"))
  (define-key menuXem08 [r-pngs]      '(menu-item "Open associated PNGs"              xem-r-pngs :help ⬋
"Open any associated PNG files"))
  (define-key menuXem08 [r-run]       '(menu-item "Invoke 'xem.job' on current model" xem-r-run  :help ⬋
"Invoke 'xem.job' call on current model without invoking 'xeona' (\"xeona = NULL\" is passed)"))

  ;; submenu 9 (reverse order)
  (define-key menuXem09 [dump-clean]  '(menu-item "Prompted en-masse clean up of current buffer" ⬋
xem-dump-clean-buffer :help "Prompted en-masse clean based on current buffer, files containing ⬋
\".xem\" remain"))
  (define-key menuXem09 [dump-unpack] '(menu-item "Select and unpack an existing tarball"        ⬋
xem-dump-unpack       :help "Select and unpack tarball into current directory, prompted for high or ⬋
low extraction (may also revert open files)"))
  (define-key menuXem09 [dump-sum]    '(menu-item "Summarize dumped models in current directory" ⬋
xem-dump-summarize    :help "Summarized recognized dumps in current directory"))
  (define-key menuXem09 [dump-xem]    '(menu-item "Dump current model to tarball"                ⬋
xem-dump              :help "Dump current model to tarball -- best with relevant R plots and HTML ⬋
visualizations"))

  ;; submenu 10 (reverse order)
  (define-key menuXem10 [ung-current] '(menu-item "Create model from current guard file"          ⬋
xem-unguard-current   :help "Create a standard model file from the current guard file"))
  (define-key menuXem10 [ung-dialog]  '(menu-item "Create model from selected guard file"         ⬋
xem-unguard-prompted  :help "Create a standard model file from a dialoged list of guard files"))
  (define-key menuXem10 [guard]       '(menu-item "Create guard file from current model"          ⬋
xem-file-guardize     :help "Create a guard file from the current model"))
  (define-key menuXem10 [bf-visits]   '(menu-item "Save various log buffers"                     ⬋
xem-buffers-visit-all :help "Save selected unvisited log buffers,the original buffer names are ⬋
retained"))
  (define-key menuXem10 [peel]        '(menu-item "Peel off a backup file"                       ⬋
xem-file-peel         :help "Peel off an incremental backup file from the current model (but consider ⬋
using RCS version control instead)"))

  ;; submenu 11 (reverse order)
  (define-key menuXem11 [sitfor]      '(menu-item "Toggle SITFOR value"                   ⬋
xem-toggle-SITFOR                :help "Toggle and then report the new value of SITFOR used in emacs ⬋
'sit-for' calls"))
  (define-key menuXem11 [repeat]      '(menu-item "Repeat last command *"                 repeat       ⬋
:help "Standard emacs 'repeat' command"))
  (define-key menuXem11 [save-some]   '(menu-item "Save some buffers *"                   ⬋
save-some-buffers                :help "Standard emacs 'save-some-buffers' command"))
  (define-key menuXem11 [kbd-quit]    '(menu-item "Quit the current emacs code (safe) *"  ⬋
keyboard-quit                    :help "Standard emacs 'keyboard-quit' command"))
  (define-key menuXem11 [line-hi]     '(menu-item "Toggle current line highlight (hcl) *" ⬋
highlight-current-line-minor-mode :help "Toggle current line highlight (by running ⬋
'highlight-current-line-minor-mode')"))
  (define-key menuXem11 [r-stub]      '(menu-item "Display world time *"                  ⬋
display-time-world               :help "Display world time"))
  (define-key menuXem11 [hi-phrase]   '(menu-item "Highlight a regex *"                   ⬋
highlight-phrase                 :help "Highlight regex phrase (by running 'highlight-phrase')"))
  (define-key menuXem11 [website]     '(menu-item "Visit xeona (deeco) website"           ⬋
xem-menu-goto-website            :help "Visit the xeona website (currently the deeco site)"))
  (define-key menuXem11 [all-dec]     '(menu-item "Decrease text size everywhere"         ⬋
xem-scale-decrease-everywhere    :help "Decrease text size everwhere"))
  (define-key menuXem11 [font-dec]    '(menu-item "Decrease text size"                    ⬋
text-scale-decrease              :help "Decrease text size using a 'text-scale' function (also ⬋
<S-down-mouse-1>)"))

 ) ; final parenthesis

;; ---------------------------------
;;  aliases
;; ---------------------------------

;; CAUTION: 'defalias' on 2007 GNU Emacs 21.4 does not support an optional DOCSTRING

;;---------------------------------
;; dot function
;;---------------------------------

(defun xem-double-dot-one ()
  "Truncate and double dot current line."
  (interactive)
```

```
  (let* ((angle  ">" )
         (line1  nil )
         (line2  nil )
         (parts  nil )
         (left   nil )
         (right  nil )
         (vals   nil )
         (val    ""  ))
    (beginning-of-line)
    (setq line1 (thing-at-point 'line))
    (delete-region (line-beginning-position) (line-end-position))  ; the newline char remains tho
    ;; split field
    (setq parts (split-string line1 angle nil))
    (setq left  (nth 0 parts))
    (setq right (nth 1 parts))                    ; could be nil
    (setq right (xem-string-trim right))          ; could be empty string
    ;; split and value
    (cond
     ;; { null }
     ((= (length right) 0) )                      ; do nothing
     ;; already dotted
     ((string-match "\\.\\.$" right) )            ; do nothing
     ;; { s S x X }
     ((string-equal (substring right 0 1) "\"")   ; leading double-quote
      (setq vals (split-string (substring right 1 -1) "\" +\"" nil))
      (setq val (nth 0 vals))
      (setq val (concat " " "\"" val "\"" " .."))
      ;; { i I f F bB } noting that boolean is handled as integer
      (t                                          ; by elimination
       (setq vals (split-string right " " nil))
       (setq val (nth 0 vals))
       (setq val (concat " " val " .."))) )
    ;; reassemble
    (setq line2 (concat left angle val))
    ;; insert
    (insert line2)
    ;; completion reporting
    (message "  xem: double-dot-one complete using:\n  %s\n %s" right val) ))

;; ---------------------------------
;;  decrease text size everywhere
;; ---------------------------------

(defun xem-scale-decrease-everywhere ()
  "Decrease text size everywhere."
  (interactive)
  (let* ((xfont "-misc-fixed-medium-r-normal--13-*-*-*-*-80-iso8859-15"))
    (set-face-attribute                       ; loaded changes propagate to new frames
     'default                                 ; so-called default face
     nil                                      ; CAUTION: 'nil' required and will propagate
     :font xfont)                             ; takes any supported "XLFD" (X logical font description)
    (message "  xem: scale-decrease-everywhere complete using '%s'." xfont) ))

;; ---------------------------------
;;  menu support files
;; ---------------------------------

(defun xem-menu-help ()
  "Wrapper to `describe-function' `xem-mode'."
  (interactive)
  (describe-function 'xem-mode))

(defun xem-menu-describe ()
  "Wrapper to `describe-mode'."
  (interactive)
  (describe-mode))

(defun xem-menu-goto-website ()
  "Goto `xem-website' value."
  (interactive)
  (message "  xem: 'xeona' website not yet written, providing 'deeco' website instead.")
  (browse-url xem-website))

(defun xem-menu-svnversion ()
  "States current svn version string."
  (interactive)
  (let* ((workingcopy  "/home/robbie/synk/xeona/svn2/futz"                          )
         (svnver       (shell-command-to-string (concat "svnversion " workingcopy)) ))
    (if (file-accessible-directory-p workingcopy)
        (progn
          (if (string-match "\n$" svnver)             ; trim trailing newline
```

```
              (setq svnver (replace-match "" t t svnver)))  ; fixed case and literal
          (message "Current svn version for '%s' is %s." workingcopy svnver))
      (message "Cannot read given working copy directory '%s'." workingcopy))))

(defun xem-menu-about ()
  "About message for `xem-mode'."
  (interactive)
  (let* ((file    (car (load-history-filename-element "xem")) )  ; name of this elisp file
         (sedstr  "s/^[[:space:]]*;\\+[[:space:]]*\\$Revision: \\([[:digit:]]*\\) \\$/\\1/p")
         (call    (concat "sed --quiet '" sedstr "' " file)   )
         (svnver  "(not set)"                                 ))
    (message "  xem: about: file: '%s'" file)
    (message "  xem: about: sed: '%s'" sedstr)
    (setq svnver (shell-command-to-string call))
    (if (string-match "\n$" svnver)                    ; trim trailing newline
        (setq svnver (replace-match "" t t svnver)))   ; fixed case and literal
    (message "  xem: about: svnver: '%s'" svnver)
    (message "%s major mode %s for editing xeona XEM model files." mode-name svnver)))

(defun xem-menu-model-status ()
  "Provide information about the current XEM file."
  (interactive)
  (let ((xem-file (buffer-file-name nil)))
    (setq xem-have-run (xem-have-i-run))     ; update status
    (if xem-have-run
        (message "Xem file '%s' has been run (but not necessarily in this session)." xem-file)
      (message "Xem file '%s' has not been run." xem-file))))

(defun xem-menu-report-binary ()
  "Provide information about the xeona binary in use."
  (interactive)
  (message "Using xeona binary '%s' with svn '%s'." xem-binary xem-svn))

(defun xem-menu-show-data-rules ()
  "Show data file rules."
  (interactive)
  (let ((call    (concat xem-binary " --data" " 2>/dev/null"))
        (capture ""))
    (switch-to-buffer xem-buffer-xeona-data-rules)
    (erase-buffer)
    (setq capture (shell-command-to-string call))
    (insert capture)
    (beginning-of-buffer)
    (message "  xem: show data rules using call '%s' complete." call)))

;; ---------------------------------
;;  SITFOR toggle
;; ---------------------------------

(defconst SITFOR 0.0)

(defun xem-toggle-SITFOR (&optional set)
  "Toggle and display `SITFOR' value, otherwise reset depending on SET.

Activate if SET is positive, deactivate if negative, and do nothing if zero"
  (interactive)
  (let* ((default  0.5    )
         (zero     0.0    )
         (prior    SITFOR ))
    (if set
        (cond
         ((< set 0)         (setq SITFOR zero   ))     ; deactivate
         ((= set 0)                               )     ; do nothing except return value
         ((> set 0)         (setq SITFOR default)))    ; activate
      (cond
       ((= prior default) (setq SITFOR zero   ))       ; toggle off
       (t                 (setq SITFOR default))))     ; toggle on
    (message "  xem: SITFOR = %f" SITFOR)
    SITFOR ))                                           ; expose return value

;; ---------------------------------
;;  file peel
;; ---------------------------------

(defun xem-file-peel ()
  "Peel off a backup file from the current model (but consider using RCS version control instead)."
  (interactive)
  (let* ((limit     98                    )  ; cycle from one to 'limit' plus one
         (peelname  (xem-file-peelname 1) ))
    (dotimes (v limit)                       ; the 'v' starts zero inclusive and ends 'limit' exclusive
      (if (file-exists-p (xem-file-peelname (+ v 1)))
```

```
          (setq peelname (xem-file-peelname (+ v 2)))))
    (if (file-exists-p peelname)
        (error "  xem: file-peel: cannot peel this file, version ceiling %d exceeded" (1+ limit)))
    (write-region nil nil peelname)          ; could also add "nil nil nil t"
    (message "  xem: file-peel: peel complete using '%s'." peelname) ))

(defun xem-file-peelname (version)
  "Create a peel filename using VERSION integer."
  (let* ((xem-file  (buffer-file-name nil)                        )  ; file with current buffer
         (xem-stub  (file-name-sans-extension xem-file)           )
         (xem-ext   (file-name-extension xem-file)                )  ; without leading dot
         (peelname  (format "%s_%02d.%s" xem-stub version xem-ext) ))
    peelname ))                              ; expose the return value

;; ---------------------------------
;;  guard file functions
;; ---------------------------------

(defun xem-file-guardize ()
  "Create and save a guard file based on the current model.

This function only processes the out-data -- the in-data is left intact.

This function does NOT rely on the various field value data-type
prompts being present and correct."
  (interactive)
  (let* ((ext        xem-model-extension                 )      ; with leading dot
         (tag        xem-model-guard-tag                 )
         (modelname  (buffer-file-name nil)              )      ; this file
         (stub       (file-name-sans-extension modelname) )
         (guardname  (concat stub tag ext)               )
         (bg         (get-buffer-create guardname)       )      ; guardfile buffer
         (count      0                                   ))
    ;; initial reporting
    (message "  xem: file-guardize: modelname '%s'" modelname)
    (message "  xem: file-guardize: guardname '%s'" guardname)
    ;; delete current guard file on request or abandon
    (if (file-exists-p guardname)
        (if (y-or-n-p "Overwrite existing guard file? ")
            (delete-file guardname)    ; TOFIX: 05-Aug-2010: should rename and delete on completion
          (kill-buffer bg)
          (error "  xem: file-guardize: guardfile '%s' exists and cannot \
be overwritten, abandoning task without action" guardname)))
    ;; continue
    (set-buffer bg)
    (set-visited-file-name guardname)
    (xem-mode)
    (insert-file modelname)
    (setq count (xem-zero-buffer bg))
    (goto-char (point-min))
    (save-buffer 0)                                ; zero means never make a backup
    (message "  xem: guardize complete using '%s', stripped %d out-data fields." guardname count)
  (message "  xem: guardize complete using '%s'." guardname) ))

(defun xem-reset-me ()
  "Reset -- zero and double-dot -- the current buffer.

Identical to calling `xem-double-dot-me' and `xem-zero-me'.

This function does NOT rely on the various field value data-type
prompts being present and correct."
  (interactive)
  (save-excursion
    (let* ((bc      (current-buffer)      )  ; lisp buffer object
           (sitfor  (xem-toggle-SITFOR 0) )
           (count1  0                     )  ; out-data changes
           (count2  0                     )) ; in-data changes
      (xem-toggle-SITFOR -1)
      (setq count1 (xem-zero-buffer bc))
      (setq count2 (xem-dotdot-buffer bc))
      (if (> sitfor 0) (xem-toggle-SITFOR +1))
      (message "  xem: reset-me: complete \
resetting %d out-data fields and double-dotting %d in-data time-series." count1 count2) )))

(defun xem-double-dot-me ()
  "Double dot the current buffer.

This function does NOT rely on the various field value data-type
prompts being present and correct."
  (interactive)
  (save-excursion
```

```
    (let* ((bc     (current-buffer) )        ; lisp buffer object
           (count  0                ))       ; in-data changes
      (setq count (xem-dotdot-buffer bc))
      (message "  xem: double-dot-me: complete double-dotting %d in-data time-series." count) )))

(defun xem-zero-me ()
  "Zero the current buffer.

This function does NOT rely on the various field value data-type
prompts being present and correct."
  (interactive)
  (save-excursion
    (let* ((bc     (current-buffer) )        ; lisp buffer object
           (count  0                ))       ; in-data changes
      (setq count (xem-zero-buffer bc))
      (message "  xem: zero-me: complete resetting %d out-data fields." count) )))

(defun xem-dotdot-buffer (buffer)
  "Double-dot the in-data time-series in BUFFER.

This function does NOT rely on the various field value data-type
prompts being present and correct."
  (save-excursion
    (let* ((angle   ">"                 )   ; indicates in-data (input data)
           (value   ""                  )
           (count   0                   )   ; number of changes
           (line1   nil                 )   ; cut
           (line2   nil                 )   ; paste
           (parts   nil                 )   ; split 'line1'
           (left    nil                 )   ; left side
           (right   nil                 )   ; right side
           (type    "(not set)"         )   ; type in { null special timeseries }
           (vals    nil                 )   ; string values
           (len     nil                 )   ; length
           (dotdot  nil                 ))  ; trailing ellipses
      ;; test buffer
      (if (not (bufferp buffer))
          (error "  xem: dotdot-buffer: given buffer failed buffer test"))
      (set-buffer buffer)                   ; make 'buffer' current for editing purposes
      (goto-char (point-min))
      ;; add initial blank line if required because later logic assumes this to be so
      (when (xem-is-field angle) (goto-char (point-min)) (insert "\n") (goto-char (point-min)))
      ;; loop in-data fields
      (while (not (eobp))
        (while (progn (forward-line +1)     ; CAUTION: order is significant
                      (xem-is-field angle))  ; CAUTION: odd logic!
          (incf count)
          (beginning-of-line)
          (setq line1 (thing-at-point 'line))
          (delete-region (line-beginning-position) (line-end-position))  ; the newline char remains tho
          ;; split field
          (setq parts (split-string line1 angle nil))
          (setq left  (nth 0 parts))
          (setq right (nth 1 parts))                    ; could be nil
          (setq right (xem-string-trim right))          ; could be empty string
          ;; split and value
          (cond
           ;; { null }
           ((= (length right) 0)                        ; empty string
            (setq len 0)
            (setq type "null"))
           ;; already dotted
           ((string-match "\\.\\.$"  right) (setq type "dotted"))
           ;; { S X }
           ((string-equal (substring right 0 1) "\"")  ; leading double-quote
            (setq vals (split-string (substring right 1 -1) "\" +\"" nil))
            (setq len (length vals))
            (cond
             ((= len 1) (setq type "single"))
             (t         (setq type "undotted"))))
           ;; { I F B } noting that boolean is handled as integer
           (t                                           ; by elimination
            (setq vals (split-string right " " nil))
            (setq len (length vals))
            (cond
             ((= len 1) (setq type "single"))
             (t         (setq type "undotted")))))
          ;; reassemble
          (cond
           ((string-match type "null"    ) (setq dotdot ""   ))
           ((string-match type "special" ) (setq dotdot ""   ))
```

```
           ((string-match type "single"  ) (setq dotdot ""   ))
           ((string-match type "dotted"  ) (setq dotdot ""   ))
           ((string-match type "undotted") (setq dotdot " .."))
           (t (error "  xem: dotdot-buffer: error: type not properly determined: %s" type)))
          (setq line2 (concat left angle " " right dotdot))
          ;; insert
          (insert line2)
          (message "  xem: dotdot-buffer: line2 : %2d %-10s : %s" len type line2)
          (sit-for SITFOR)
          )) ; 'while' x 2
      ;; completion reporting
      (message "  xem: dotdot-buffer: complete, %d values removed or modified." count)
      count )))

(defconst xem-empty-out-fields
  (list "process-id" "run-kind" "used-svn" "simulate-return")
  "List of out-data field to remain strictly empty.")

(defun xem-zero-buffer (buffer)
  "Zero BUFFER.

This function does NOT rely on the various field value data-type
prompts being present and correct."
  (save-excursion
    (let* ((angle   "<"                   )   ; indicates out-data (output data)
           (emptys  xem-empty-out-fields )
           (value   ""                   )
           (count   0                    )   ; number of changes
           (line1   nil                  )   ; cut
           (line2   nil                  )   ; paste
           (parts   nil                  )   ; split 'line1'
           (left    nil                  )   ; left side
           (right   nil                  )   ; right side
           (type    nil                  )   ; type in { null special string int float }
           (vals    nil                  )   ; string values
           (val     nil                  )   ; first 'vals'
           (nums    nil                  )   ; number values
           (num     nil                  )   ; first 'nums'
           (len     nil                  ))  ; length
      ;; teste buffer
      (if (not (bufferp buffer))
          (error "  xem: zero-buffer:: given buffer failed buffer test"))
      (set-buffer buffer)                ; make 'buffer' current for editing purposes
      (goto-char (point-min))
      ;; add initial blank line if required because later logic assumes this to be so
      (when (xem-is-field angle) (goto-char (point-min)) (insert "\n") (goto-char (point-min)))
      ;; loop out-data fields
      (while (not (eobp))
        (while (progn (forward-line +1)      ; CAUTION: order is significant
                      (xem-is-field angle))  ; CAUTION: odd logic!
          (incf count)
          (beginning-of-line)
          (setq line1 (thing-at-point 'line))
          (delete-region (line-beginning-position) (line-end-position))  ; the newline char remains tho
          ;; split field
          (setq parts (split-string line1 angle nil))
          (setq left  (nth 0 parts))
          (setq right (nth 1 parts))                    ; could be nil
          (setq right (xem-string-trim right))          ; could be empty string
          ;; split and value
          (cond
           ;; { null }
           ((= (length right) 0)                        ; empty string
            (setq len 0)
            (setq value "")
            (setq type "null"))
           ;; { s S x X } with x, being a character stream, is essentially a string
           ((string-equal (substring right 0 1) "\"")  ; leading double-quote
            (setq vals (split-string (substring right 1 -1) "\" +\"" nil))
            (setq val (nth 0 vals))
            (setq len  (length vals))
            (setq value " \"\"")
            (if (> len 1) (setq value (concat value " ..")))
            (setq type "string"))
           ;; { i I f F b B } noting that boolean is handled as integer
           (t                                           ; by elimination
            (setq vals (split-string right " " nil))
            (setq val (nth 0 vals))
            (setq len (length vals))
            (setq num  (string-to-number val))
            (cond
```

```
             ((integerp num) (setq type "int"  ) (setq value " 0"  ))
             ((numberp  num) (setq type "float") (setq value " 0.0"))
             (t (message "  xem: zero-buffer: problem: type not identified, right %s" right)))
            (if (> len 1) (setq value (concat value " .."))))))
          ;; special treatment for the program fields
          (dolist (empty emptys)
            (setq regex (concat "" empty ""))
            (when (string-match regex left)
              (setq type "special")
              (setq value "")))
          ;; reassemble
          (setq line2 (concat left angle value))
          ;; insert
          (insert line2)
          (message "  xem: zero-buffer: line2 : %2d %-7s : %s" len type line2)
          (sit-for SITFOR)
          )) ; 'while' x 2
      ;; completion reporting
      (message "  xem: zero-buffer: complete, %d values removed or modified." count)
      count )))

(defun xem-strip-field-data (buffer angle)
  "Strip field data from buffer BUFFER using ANGLE.

BUFFER is tested for existence.  ANGLE must be either \">\" for
in-data or \"<\" for out-data.  If not, an error is generated."
  (let ((count  0 ))                        ; strip count
    (if (not (string-match "^[<>]$" angle))
        (error "  xem: strip-field-data: invalid ANGLE argument '%s' supplied (coding error)" angle))
    (if (not (bufferp buffer))
        (error "  xem: strip-field-data: given buffer failed buffer test"))
    (set-buffer buffer)                     ; make 'buffer' current for editing purposes
    (goto-char (point-min))
    ;; add initial blank line if required because later logic assumes this to be so
    (when (xem-is-field angle) (goto-char (point-min)) (insert "\n") (goto-char (point-min)))
    ;; loop out-data fields
    (while (not (eobp))
      (while (progn (forward-line +1)       ; CAUTION: order is significant
                    (xem-is-field angle))   ; CAUTION: odd logic!
        (incf count)
        (beginning-of-line)
        (setq line (thing-at-point 'line))
        (delete-region (line-beginning-position) (line-end-position))  ; the newline char remains tho
        (setq parts (split-string line angle nil))
        (insert (nth 0 parts))
        (insert angle)
        (message "  xem: strip-field-data: count %2d line '%s'" count line)
        (sit-for SITFOR)
        )) ; 'while' x 2
    (message "  xem: strip-field-data: buffer stripped, %d values removed using angle '%s'"
             count angle)
    count ))

;; ---------------------------------
;;  hilighter functions
;; ---------------------------------

;; inspiration: markerpen.el --- colour and highlight arbitrary sections of buffers

(defun xem-hilighter-entity ()
  "Hilight (apply color to) the current xeona entity."
  ;; a somewhat smart interface to `xem-hilighter-region'
  (interactive)
  (save-excursion
    (let* ((entity-end  (1- (xem-move-entity-next)) )  ; move down first, note the small adjustment
           (entity-beg  (xem-move-entity-prior)     ))
      (if (and entity-beg entity-end)
          (progn
            (xem-hilighter-region entity-beg entity-end)
            (message "  xem: hilighter entity complete using %d:%d" entity-beg entity-end))
        (message "  xem: hilighter entity unable to identify a current entity")) )))

;; hilighter colors

(defface xem-hilighter-a '((t :background "dark sea green")) "hilighter a")     ; vivid
(defface xem-hilighter-b '((t :background "khaki3"        )) "hilighter b")     ; tertiary
(defface xem-hilighter-c '((t :background "peru"          )) "hilighter c")     ; dark but subtle
(defface xem-hilighter-d '((t :background "beige"         )) "hilighter d")     ; same as entity label

;; hilighter support
```

```
(make-variable-buffer-local 'xem-hilighter-overlays)

(defun xem-hilighter-region (reg-beg reg-end)
  "Hilight (apply color to) the current region ranging REG-BEG thru REG-END.

The hilighters are implemented using overlays, so they do not, in
any way, affect the contents of the buffer -- even if the buffer
uses text properties.  Because of this you can quite happily use
hilighters together with a mode which uses font locking.

The function `xem-hilighter-clear-buffer' removes all marks."
  (interactive "r")                         ; indicates region, no i/o
  (let ()
    (setq new-hilighter-overlay (make-overlay reg-beg reg-end))
    (deactivate-mark)                       ; for transient-mark-mode
    (add-to-list 'xem-hilighter-overlays new-hilighter-overlay)
    (overlay-put new-hilighter-overlay 'face 'xem-hilighter-d) ))     ; change color here

(defun xem-hilighter-clear-buffer ()
  "Clear all hilights from the current buffer."
  (interactive)
  (mapcar 'xem-hilighter-delete-overlay xem-hilighter-overlays)
  (message "  xem: hilight clear buffer complete."))

(defun xem-hilighter-clear-region (reg-beg reg-end)
  "Clear all hilights from the current region ranging REG-BEG thru REG-END."
  (interactive "r")
  (if (use-region-p)
      (progn
        (mapcar 'xem-hilighter-delete-overlay (overlays-in reg-beg reg-end))
        (deactivate-mark)                   ; for transient-mark-mode
        (message "  xem: hilight clear region complete."))
    (message "  xem: hilight clear region complete but no region was active.") ))

(defun xem-hilighter-delete-overlay (overlay)
  "Delete the supplied OVERLAY if it is a hilighter overlay."
  (setq xem-hilighter-overlays (delq overlay xem-hilighter-overlays))
  (delete-overlay overlay))

;; ---------------------------------
;;  overwrite simulate-return
;; ---------------------------------

; the form is: >    simulate-return                              < "success (1)"<

(defun xem-fix-incomplete-run (&optional return)
  "Overwrite the 'simulate-return' field when xeona fails to write the model.  Add RETURN if given."
  (save-excursion
    (let ((fieldname  "simulate-return"               )
          (splif      "<"                             )
          (tag1       "model failed before write"     )
          (tag2       "xeona exit"                    )
          (msg1       nil                             )
          (msg2       nil                             )
          (line       nil                             ))
      ;; create message strings
      (setq msg1 tag1)
      (if (not return)
          (setq msg2 tag2)                  ; but the 'return' should normally be known
        (if (integerp return)
            (setq msg2 (format "%s %d" tag2 return))
          (setq msg2 (format "%s %s" tag2 return))))
      ;; insert message
      (goto-char (point-min))
      (if (re-search-forward fieldname nil t)
          (progn
            (setq line (thing-at-point 'line))
            (delete-region (line-beginning-position) (line-end-position))
            (setq parts (split-string line splif nil))
            (insert (nth 0 parts))
            (insert (concat splif " \"" "** " msg1 " **" "\"" "\n"))  ; trailing newline
            (insert (concat "\n" "    " "** " msg2 " **"))            ; skip a line first
            (message "  xem: fix-incomplete-run: complete using '%s'." return))
        (message "  xem: fix-incomplete-run: unable to locate fieldname '%s'." fieldname)) )))

;; ---------------------------------
;;  yeek and exit code information
;; ---------------------------------

; typical yeek:    10 = report from 'xeona::isTwoContained' call
```

```
(defun xem-get-yeek (yeek)
  "Return yeek explanation associated with YEEK."
  (interactive)
  (let ((record  12))                       ; record 12, may change but unlikely
    (xem-get-info record yeek)))

(defun xem-get-code (code)
  "Return exit code explanation associated with CODE."
  (interactive)
  (let ((record  13))                       ; record 13, may change but unlikely
    (xem-get-info record code)))

(defun xem-get-info (record number)
  "Return explanation from RECORD associated with NUMBER."
  (let* ((call-1   (concat xem-binary " --usage 2>/dev/null") )
         (call-2   (format "%s%d%s" "awk 'BEGIN { RS = \"\\n\\n\"} { if ( NR == "
                           record
                           " ) print $0 }' | tail --lines=+2" ) )
         (call     (concat call-1 " | " call-2)   )
         (capture  (shell-command-to-string call) )
         (lines    (split-string capture "\n" t)  )  ; 't' is omit nulls ('nil' makes no difference)
         (regex    nil                            )
         (explain  nil                            ))
    (if (integerp number)
        (progn
          (setq regex (format "^[[:space:]]*%d = \\(.*\\)$" number))
          (message "  xem: get-yeek: regex %s" regex)
          (dolist (line lines)
            (if (string-match regex line)
                (setq explain (match-string 1 line))))
          (message "  xem: get-yeek: explain %s" explain)
          explain)                          ; return the explanation
      (error "  xem: get-yeek: number not integer")) ))

;; ---------------------------------
;;  prioritize coloration
;; ---------------------------------

;; prioritize coloration over current line highlighting
;; source: http://www.emacswiki.org/emacs/HiLock
;; status: works as advertised up to about 2000 lines

(defadvice hi-lock-set-pattern (around use-overlays activate)
  "Function `hi-lock-mode' should prioritize coloration over current line highlighting."
  (let ((font-lock-fontified nil))
    ad-do-it))

;; ---------------------------------
;;  buffer switch functionality
;; ---------------------------------

;; key binding (see above) is common with 'xeona.el'

(defun xem-switch-to-useful-buffer ()
  "Switch to useful buffer."
  (interactive)
  (let ((target  (get-buffer xem-buffer-xeona-output) ))
    (if (and (not (null target))                ; evaluated in order written
             (not (eq (current-buffer) target)))  ; 'eq' for same lisp object
        (switch-to-buffer target)
      (list-buffers)
      (other-window)) ))

;; ---------------------------------
;;  guard file processing
;; ---------------------------------

;; menu item: "Create model from selected guard"
;; temporary while developing ad-hoc models

(defun xem-unguard (guardname)
  "Unguard the GUARDNAME file."
  (let* ((guard    guardname                          )
         (unguard  (xem-unguard-filename guardname) )) ; possibly "stub.xem", possibly nil
    (if unguard                                       ; name recovery worked
        (if (file-exists-p guard)
            (progn
              (copy-file guard unguard 1)    ; "1" is seek confirmation on overwrite - shouldn't happen
              (find-file unguard)
              (message "  xem: futz processed guard file '%s'." guard))
          (message "  xem: unguard abandoning task as guard file '%s' not found." guard))
```

```
      (message "  xem: unguard abandoning task as selected file '%s' is not a guard file." guard)) ))

(setq use-file-dialog t)                    ; 't' is use mouse dialog from drop-down menu

(defun xem-unguard-current ()
  "Unguard the current file.

A buffer file name interface to `xem-unguard'."
  (interactive)
  (let ((guardname  (buffer-file-name nil) ))     ; this file, possibly "stub.guard.xem"
    (xem-unguard guardname) ))

(defun xem-unguard-prompted (guardname)
  "Unguard the prompted GUARDNAME file.

An ask-for-filename interface to `xem-unguard'."
  (interactive "f")
  (xem-unguard guardname))

(defun xem-unguard-filename (filename)
  "Unguard FILENAME, also acts as a predicate."
  (let* ((model-ext  (substring xem-model-extension 1)   )  ; now "xem"
         (guard-tag  (substring xem-model-guard-tag 1)   )  ; now "guard"
         (str-1      (file-name-sans-extension filename) )  ; attempt to trim ".xem"
         (str-2      (file-name-sans-extension str-1)    )
         (str-3      (file-name-extension str-1)         )) ; may return "guard", also "" or nil
    (if (and (not (null str-3))                             ; evaluated in order written
             (string-match str-3 guard-tag))
        (progn
          (message "  xem: unguard-file: true: %s." str-3)
          (concat str-2 "." model-ext))                     ; return unguarded name
      (message "  xem: unguard-file: false: %s." str-3)
      nil) ))                                               ; return nil (previously an empty string)

;; ---------------------------------
;;  screen modes
;; ---------------------------------

;; CAUTION: the toggle semantics here differ from the
;; usual toggle behavior for emacs

(defun xem-fullscreen-me-toggle ()
  "Toggle full-screen mode."
  (interactive)
  (xem-fullscreen-me t))

(defun xem-fullscreen-me (&optional toggle)
  "Adopt full-screen mode.  If TOGGLE is t, then alternate with part-screen mode."
  ;; CAUTION: contains protected platform-specific code
  ;; http://www.emacswiki.org/emacs/FullScreen
  (cond
   ((eq system-type 'gnu/linux)
    ;; 1 = fixed full, 2 = toggle for each call
    (if toggle
        (x-send-client-message nil 0 nil "_NET_WM_STATE" 32 '(2 "_NET_WM_STATE_FULLSCREEN" 0))
      (x-send-client-message   nil 0 nil "_NET_WM_STATE" 32 '(1 "_NET_WM_STATE_FULLSCREEN" 0))))
   ((eq system-type 'windows-nt)
    (error "  xem: full-screen mode not coded for Windows operating system"))
   (t
    (error "  xem: full-screen mode not coded for this operating system"))) )

(defun xem-maxscreen-me-toggle ()
  "Toggle maximize mode."
  (interactive)
  (xem-maxscreen-me t))

(defun xem-maxscreen-me (&optional toggle)
  "Adopt maximize mode.  If TOGGLE is t, then alternate with part-screen mode."
  ;; CAUTION: contains protected platform-specific code
  ;; http://www.emacswiki.org/emacs/FullScreen
  (cond
   ((eq system-type 'gnu/linux)
    ;; 1 = fixed full, 2 = toggle for each call
    (if toggle
        (progn
          (x-send-client-message nil 0 nil "_NET_WM_STATE" 32 '(2 "_NET_WM_STATE_MAXIMIZED_VERT" 0))
          (x-send-client-message nil 0 nil "_NET_WM_STATE" 32 '(2 "_NET_WM_STATE_MAXIMIZED_HORZ" 0)))
      (x-send-client-message   nil 0 nil "_NET_WM_STATE" 32 '(1 "_NET_WM_STATE_MAXIMIZED_VERT" 0))
      (x-send-client-message   nil 0 nil "_NET_WM_STATE" 32 '(1 "_NET_WM_STATE_MAXIMIZED_HORZ" 0))))
   ((eq system-type 'windows-nt)
    (error "  xem: maximize mode not coded for Windows operating system"))
```

```
   (t
    (error "  xem: maximize mode not coded for this operating system"))) )

;; ---------------------------------
;;  diff output functions
;; ---------------------------------

(defconst xem-default-face-height (face-attribute 'default :height) "Current font size.")
(defconst xem-small-face-height   90                                "Small font size.")

;; 'ediff' code
;; http://braeburn.aquamacs.org/code/master/lisp/ediff.el
;; http://www.emacswiki.org/emacs/SetFonts
;; note also <S-down-mouse-1> for interactive "Change Default Buffer Face" menu

(add-hook 'ediff-before-setup-hook 'xem-ediff-entry)
(add-hook 'ediff-cleanup-hook      'xem-ediff-exit)    ; also 'ediff-quit-hook'

(defun xem-ediff-entry ()
  "Xem 'ediff' entry function."
  (let ((face-height  xem-small-face-height ))
    (message "  xem: ediff-entry: commencing")
    (setq ediff-split-window-function 'split-window-horizontally)     ; ediff side-by-side
;;; (setq ediff-keep-variants nil)                                    ; ediff close created windows
    (set-face-attribute 'default (selected-frame) :height face-height)
    (xem-fullscreen-me nil)                                           ; CAUTION: [1]
    (message "  xem: ediff-entry: complete") ))
    ;; [1] fixed full-screen must come after face height

(defun xem-ediff-exit ()
  "Xem 'ediff' exit function."
  ;; CAUTION: this function is sensitive to the ordering of statements
  (let ((face-height  xem-default-face-height )
        (buffer-a     ediff-buffer-A          )   ; note also 'ediff-window-A' etc
        (buffer-b     ediff-buffer-B          )
        (quick-help   ediff-control-frame     ))  ; CAUTION: frame details must be captured early on
    (message "  xem: ediff-exit: commencing")
    (message "  xem: ediff-exit: buffers: A B %s %s" buffer-a buffer-b)
    (message "  xem: ediff-exit: quick-help: %S" quick-help)
    (kill-buffer      buffer-b)
    (switch-to-buffer buffer-a)
    (delete-other-windows)
    (set-face-attribute 'default nil :height face-height)   ; 'nil' is reset all
    (xem-fullscreen-me t)                                   ; CAUTION: [1]
    (if quick-help (delete-frame quick-help))               ; nuke the little quick help frame
    (message "  xem: ediff-exit: complete") ))
    ;; [1] fixed full-screen must come after face height

(defun xem-diff-me-ediff ()
  "Ediff current model after setting face height and adopting full-screen mode."
  (interactive)
  (xem-diff-me ?e))

(defun xem-diff-me (type)
  "Diff current model in new buffer, based on TYPE key press.

    e    ediff
    u    unix diff    (command-line unix diff)
    ?    help         (this message)"
  (interactive "cType character for diff type ([e]diff unix-diff)? ")
  (let* ((diff-switches    ""                                  )
         (diff-buffer      xem-buffer-xem-diff                 )
         (backup-tag       xem-backup-tag                      )
         (xem-file-1       (buffer-file-name nil)              )
         (xem-file-2       (concat xem-file-1 backup-tag)      )
         (xem-leaf-1       (file-name-nondirectory xem-file-1) )
         (xem-face-height  0                                   ))
    (if (file-readable-p xem-file-2)
        (progn
          (message "  xem: diff-buffer: type %c" type)
          (if (equal type 13)               ; catch return key ^M
              (setq type ?e))               ; set default here
          (cond                             ; elisp case statement
           ;; ediff, see also: http://www.emacswiki.org/emacs/EdiffMode
           ((equal type ?e)
            (ediff xem-file-1 xem-file-2))
           ;; diff(1), see diff manpage
           ((equal type ?u)
            (diff xem-file-1 xem-file-2 diff-switches)
            (other-window +1)
            (rename-buffer diff-buffer)
```

```
            (other-window -1))

           ((equal type ??)
            (describe-function 'xem-diff-me))
           (t
            (error "  xem: diff me quitting without action, character '%c' not supported" type)))
          (message "  xem: diff me complete using '%s' and backup tag '%s'." xem-file-1 backup-tag))
      (message "  xem: diff me abandoned as backup file '%s' not present or not readable." xem-file-2))
    ))

;; ---------------------------------
;;  clean model functions
;; ---------------------------------

(defun xem-clean-model ()
  "Clean model of \"(some problem)\" entries.

Largely obsolete, but could be useful in combination with --yeek 33"
  (interactive)
  (save-excursion
    (let ((find   "\"(some problem)\"")
          (repl   "\"\""              )
          (count 0))                        ; replacements made
      (goto-char (point-min))               ; beginning of buffer
      (while (re-search-forward find nil t)
        (replace-match repl t t)
        (incf count))                       ; increment count
      (message "  xem: clean model complete using '%s' with %d replacements." find count))))

;; ---------------------------------
;;  run model and related functions
;; ---------------------------------

(defun xem-get-field-value (fieldname)
  "Return the value associated with FIELDNAME.  Else return nil."
  ;; comment: currently hard-coded for input (<) fields but that could be easily changed
  (save-excursion
    (let ((line  nil ))                     ; field line, if present
      (goto-char (point-min))
      ;; locate fieldname, grab line, strip newline, and return
      (if (re-search-forward fieldname)
          (progn
            (setq line (thing-at-point 'line))
            (setq line (xem-string-trim line))
            (message "  xem: get-field: line found '%s'" line)
            (xem-extract-data line fieldname ">"))
        (message "  xem: get-field: not found (returning nil)")
        nil) )))

(defun xem-run-me (numbers)
  "Run xeona using NUMBERS to code for 'report|mode|yeek' values.

This function provides a pass-thru interface to
'xeona-run-me-call'.  The various system calls and buffer and
file modification commands are located in 'xeona-run-me-call'.

This function normally recovers and passes on any sensible
'program.run-script-settings' options.  A negative sign means
omit this feature.

Out of range values are checked.  High values for 'report' and 'mode' are
reset as required.  An invalid 'yeek' value results in abandonment."
  (interactive "nEnter report [ mode [ yeek ]]: ")
  (let ((noas      ""   )                   ; number as string
        (said      ""   )                   ; report
        (mode      "7"  )                   ; default mode
        (yeek      "0"  )                   ; default yeek
        (maxyeek   40   )                   ; current maximum yeek value (can change)
        (options   nil  )                   ; options
        (output    ""   ))                  ; final output
    ;; if negative, process any 'program.run-script-settings' options
    (if (< numbers 0)
        (message "  xem: run-me: negative")
      (setq exittrip (xem-get-field-value "script-option-exittrip"))
      (setq nodata   (xem-get-field-value "script-option-nodata"))
      (setq jumpy    (xem-get-field-value "script-option-jumpy"))
      (if exittrip (push (list (concat "--exittrip " exittrip)) options))
      (if nodata (if (not (string-equal nodata "0")) (push (list "--nodata") options)))
      (if jumpy  (if (not (string-equal jumpy  "0")) (push (list "--jumpy" ) options))))
    ;; process implied options
    (setq noas (format "%d" (abs numbers)))
```

```
    (setq len (length noas))
    (message "  xem: run-me: numbers %d" numbers)
    (message "  xem: run-me: length %d" len)
    (if (>= len 1) (setq said (substring noas 0   1)))
    (if (>= len 2) (setq mode (substring noas 1   2)))
    (if (>= len 3) (setq yeek (substring noas 2 len)))      ; checked later against current 'maxyeek'
    (if (>  (string-to-number said) 7) (setq said  "7"))    ; maximum '--report' value
    (if (>  (string-to-number mode) 7) (setq mode  "7"))
    (if (> (string-to-number yeek) maxyeek)
        (error "  xem: run me refusing to run with yeek %s as it exceeds %d" yeek maxyeek))
    (push (list (concat "--report " said)) options)
    (push (list (concat "--mode "   mode)) options)
    (push (list (concat "--yeek "   yeek)) options)
    (setq output
          (with-output-to-string
            (dolist (option (reverse options))
              (princ (car option))
              (princ " "))))
    (setq output (xem-string-trim output))   ; trim last space
    (message "  xem: run-me: options %s" output)
    (xem-run-me-call output) ))

(defun xem-run-me-call (options)
  "Call `xem-run-me-call' using given OPTIONS."
  ;; file abbreviation (conversion from "$HOME/$USER" to "~") has no effect on 'xeona' logging
  (let* ((out-buffer    xem-buffer-xeona-output             )
         (bm            (current-buffer)                    )    ; xem file buffer
         (bo            (get-buffer-create out-buffer)      )    ; output capture buffer
         (xem-file      (buffer-file-name bm)               )
         (xem-leaf      (file-name-nondirectory xem-file)   )
         (temp          (split-string options)              )
         (yeekarg       (car (last temp))                   )    ; rather brittle!
         (yeek          (string-to-number yeekarg)          )
         (linemsg       nil                                 )
         (yeekmsg       nil                                 )
         (exitmsg       nil                                 )
         (filemsg       nil                                 )
         (call          ""                                  )
         (code          0                                   )    ; exit code
         (timestamp-1   nil                                 )
         (timestamp-2   nil                                 )
         (update        "(not set)"                         )
         (vhold         0                                   )    ; vertical (line) hold position
         (hhold         0                                   ))   ; horizontal (column) hold position
    ;; reporting
    (message "  xem: run-me-call: binary   %s" xem-binary)
    (message "  xem: run-me-call: options  %s" options)
    (message "  xem: run-me-call: xem file %s" xem-file)
    ;; confirm not a guard file, then check write permission
    (if (xem-unguard-filename xem-file)
        (message "  xem: run me call given guard file '%s', abandoning task without action."
                 xem-file)
      (if (not (file-writable-p xem-file))
          (message "  xem: run me call given read-only file '%s', abandoning task without action."
                   xem-file)
        ;; create call
        (setq call (concat xem-binary " " options " --file " xem-file))
        ;; prompted save if required ('bm' assumed here)
        (if (buffer-modified-p)
            (if (y-or-n-p "Save this model in order to continue? ")
                (save-buffer)
              (error "  xem: run me call abandoning '%s' call without action" call)))
        (save-buffer)                                       ; defensive programming, not really needed
        ;; record xem buffer
        (setq vhold (+ (count-lines (point-min) (point))    ; "1" indicates the first line
                       (if (= (current-column) 0) 1 0)))    ; correction for start of line, note:
        (setq hhold (current-column))                       ; "0" indicates the start-of-line column
        (message "  xem: run-me-call: vhold hhold: %d %d" vhold hhold)
        (setq timestamp-1 (xem-file-modified xem-file))     ; capture last modified
        (sleep-for 1)                                       ; to ensure distinct last modified times
        ;; prepare output buffer
        (set-buffer bo)                                     ; 'bo' is the output buffer
        (erase-buffer)                                      ; delete entire contents, ignores narrowing
        ;; make call
        (setq code (shell-command call bo nil))             ; 'nil' means mingle stdout and stderr
        (message "  xem: run-me-call: xeona ran synchronously and returned '%s'" code)
        ;; rework output buffer
        (xog-mode)                                          ; xeona output mode
        (toggle-truncate-lines 1)                           ; "1" is never fold (wrap) long lines
        (delete-trailing-whitespace)                        ; may (and probably does) modify the buffer
        (end-of-buffer)                                     ; jump to end
```

```
        ;; reload xem file
        (switch-to-buffer bm)
        (delete-other-windows)                              ; disable to keep both buffers visible
        (revert-buffer t t t)                               ; 'ignore-auto''noconfirm''preserve-modes'
        (setq timestamp-2 (xem-file-modified xem-file))     ; capture last modified
        (if (equal timestamp-1 timestamp-2)                 ; [1]
            (progn
              (setq update "1 = UNMODIFIED")
              (xem-fix-incomplete-run code))
          (setq update "0 = modified"))
        (goto-line vhold)                                   ; CAUTION: must precede next line
        (move-to-column hhold)
        (message "  xem: run-me-call: xem file revisited: xem-file")
        ;; completion reporting
        (setq linemsg (format "file '%s' with options '%s'" xem-leaf options))
        (setq yeekmsg (format "yeek  %2d = %s" yeek (xem-get-yeek yeek)))
        (if (integerp code)                                 ; can be "Aborted"
            (setq exitmsg (format "exit  %2d = %s" code (xem-get-code code)))
          (setq exitmsg (format "exit  '%s' (from shell not xeona)" code)))
        (setq filemsg (format "model  %s" update))
        (message "  xem: run me with %s\n       %s\n       %s\n       %s"
                 linemsg yeekmsg exitmsg filemsg))) ))
        ;; [1] tried (verify-visited-file-modtime) to no avail

(defun xem-file-modified (filename)
  "Return last modified timestamp for FILENAME.

More specifically, return the fifth field from 'file-attributes', a list
of two integers, which represents the timestamp.

Typical output (19516 57041) as the fourth element of 'file-attributes.'
Typical output (57 46 23 13 7 2010 2 t 7200) under 'decode-time.'"
  (let* ((attribs (file-attributes filename 'string) )
         (timestamp (nth 4 attribs)                  ))     ; zero-based list indexing
    (message "  xem: file-modified complete using '%s' and giving %S and %s."
             filename timestamp (decode-time timestamp))
    timestamp))                                             ; return timestamp

(defun xem-run-tidy ()
  "Run xeona '--tidy' on current buffer.

Calls 'xem-run-me-call' using option '--tidy'."
  (interactive)
  (let ((options "--tidy"))
    (xem-run-me-call options)))

(defconst xem-run-me-call-regex              ; order counts
  "--[-=[:alnum:]]+[[:space:]]arg\\|--[-=[:alnum:]]+[[:space:]][+[:digit:]]+\\|--[-=[:alnum:]]+")

(defun xem-usage-filtered (filter)
  "Filter xeona '--usage' output based on FILTER regex.

Calls 'xem-run-me-call' using option '--usage' and then applies 'string-match'."
  (interactive "sEnter a regex usage filter (or + for all) ")
  (let* ((buffer   xem-buffer-xeona-usage                    )
         (call     (concat xem-binary " --usage 2>/dev/null") )
         (capture  (shell-command-to-string call)             )
         (lines    (split-string capture "\n" t)              )  ; 't' is omit nulls ('nil' also works)
         (regex    xem-run-me-call-regex                      )
         (hits     ""                                         ))
    (cond ((string-equal filter "+")
           (switch-to-buffer buffer)
           (insert capture)
           (delete-trailing-whitespace)
           (goto-char (point-min))
           (highlight-phrase regex 'xem-hi-color-B07)
           (message "  xem: full usage complete using '%s'." filter))
          (t
           (dolist (line lines)
             (if (string-match filter line)
                 (setq hits (concat hits line "\n"))))
           (if (string-match "\n$" hits)                    ; trim trailing newline
               (setq hits (replace-match "" t t hits)))   ; fixed case and literal
           (message "  xem: filtered usage using regex'%s':\n%s" filter hits))) ))

;; --------------------------------
;;  buffer functions
;; --------------------------------

;; interactive function 'xem-revisit' is provided as a menu item,
;; however (revert-buffer t t t) may be equivalent and better
```

```
;; style -- note too that, at the time of writing, the
;; 'Global-Auto-Revert' mode was automatically activated

(defun xem-revisit ()
  "Revist current file."
  (interactive)
  (let ((xem-file (buffer-file-name nil)))   ; this file
    (save-buffer)                           ; emacs will ask if write permissions have been withdrawn
    (kill-buffer)
    (find-file xem-file)
    (message "  xem: revisit complete using '%s'." (abbreviate-file-name xem-file))))

(defun xem-insert-strong-comment ()
  "Insert \"** COMMENT\", suitably aligned."
  (interactive)
  (let* ((prepend  "** "                 )
         (marker   "COMMENT"             )
         (line     (thing-at-point 'line) )
         (len      (string-bytes line)    ))
    (when (> len 1)                         ; contains only a newline if empty
      (end-of-line)
      (insert "\n"))
    (insert "      ")                       ; 6 spaces
    (insert prepend)
    (insert marker)
    (message "  xem: insert string comment complete.")))

;; ---------------------------------
;;  find binary
;; ---------------------------------

(defvar xem-binary   nil "Binary name.")
(defvar xem-have-run nil "XEM file have run flag.")    ; 'nil' = not run, 't' = run
(defvar xem-svn      nil "Binary svn revision.")       ; should be zero or integer

(make-local-variable 'xem-binary)
(make-local-variable 'xem-have-run)
(make-local-variable 'xem-svn)

(defun xem-rebinary (new-binary)
  "Prompted find for new xeona binary NEW-BINARY."
  (interactive "f")
  (let* ((old-binary  xem-binary))
    (setq xem-binary new-binary)
    (setq xem-svn (xem-binary-svn xem-binary))
    (message " xem: rebinary complete using '%s',\n                   previously '%s'."
             xem-binary old-binary) ))

(defun xem-locate-binary (&optional xeona) ; optional binary name
  "Locate the desired XEONA binary somewhat intelligently.

Indeed it can be useful during testing not to place the 'xeona' binary
on the shell search path."
  (interactive)
  (let ((binary  "xeona.mach"                          )     ; default binary name
        (paths   (list "." ".." "../.." "../xeona1" ) )     ; preferred search order
        (which   nil                                   )     ; which output
        (call    nil                                   ))    ; return value
    (if xeona (setq binary xeona))           ; overwrite the default as required
    (setq which (shell-command-to-string (concat "which" " " binary)))
    (if (> (length which) 0)
        ;; 'binary' is on the shell search PATH
        (setq call (xem-string-trim which))  ; remove the trailing newline
      ;; else hunt thru the given list of 'paths'
      (setq paths (nreverse paths))          ; work backwards
      (dolist (path paths)
        (setq possible (expand-file-name binary path))
        (if (file-executable-p possible)
            (setq call possible))))
    ;; report
    (if call
        (message "  xem: locate-binary call '%s'" call)
      (message "  xem: locate-binary FAIL using binary name '%s'" binary))
    call))                                   ; expose the return value

(defun xem-binary-svn (binary)
  "Obtain svn revision of named BINARY."
  (let ((capture  nil ))                     ; shell command capture
    (if binary
        (progn
          (setq capture (shell-command-to-string (concat binary " --svn" " 2>/dev/null")))
```

```
          (setq capture (xem-string-trim capture))
          (message "  xem: xem-binary-svn: capture '%s'" capture)
          (string-to-number capture))
      (message "  xem: xem-binary-svn: binary set to nil")
      nil)))

;; ---------------------------------
;;  create new model
;; ---------------------------------

(defun xem-insert-new-model (&optional xeona)
  "Insert the output from \"--xem comb\" using the XEONA binary, if supplied."
  (interactive)
  (let ((option  "--xem comb" )              ; xeona option
        (skel    ""           ))             ; skeleton XEM
    (if (not xeona)                          ; xeona name not set externally
        (setq  xeona xem-binary))
    (if (file-executable-p xeona)            ; xeona is executable by you
        ;; main code
        (progn
          (if (> (point-max) (point-min))
              (if (y-or-n-p "Buffer not empty, continue? ")
                  ()
                (error "  xem: insert-new-model: abandoning task without action on user request")))
          (setq call (concat xeona " " option " " "2>/dev/null"))
          (setq skel (shell-command-to-string call))
          (beginning-of-line)
          (insert skel)
          ;; completion reporting
          (message "  xem: insert-new-model complete."))
      (progn
        ;; executable not found reporting
        (message "  xem: insert-new-model FAILURE, xeona executable not found '%s'." xeona)))))

;; ---------------------------------
;;  rule shift
;; ---------------------------------

(defvar xem-rule-indent 2 "Current rule indent value, usually either 1 or 2.")
(make-local-variable 'xem-rule-indent)

(defun xem-toggle-rules ()
  "Toggle rule indentation between 1 and 2 for reasons of visibility."
  (interactive)
  (save-excursion                            ; hold cursor position
    (let ((rule  "----------" ))             ; used to identify rules
      ;; toggle
      (cond ((= xem-rule-indent 1) (setq xem-rule-indent 2))
            ((= xem-rule-indent 2) (setq xem-rule-indent 1))
            (t (message "  xem: toggle-rules: cond fall-thru")))
      ;; action
      (goto-char (point-min))
      (while (not (eobp))
        (if (looking-at (concat "^[[:blank:]]+" rule))
            (xem-rule xem-rule-indent))      ; the argument is strictly unnecessary
        (forward-line))
      (message "  xem: toggle-rules complete with %d" xem-rule-indent)) ))

;; ---------------------------------
;;  highlight output
;; ---------------------------------

(defvar xem-highlight-output-flag nil
  "Non-nil means `xem-toggle-highlight-output' active.")
(make-local-variable 'xem-highlight-output-flag)

(defun xem-toggle-highlight-output ()
  "Toggle Xem field 'program.r-processing.r-highlight-output' highlighting.

Utilizes `occur'."
  (interactive)
  (if (xem-have-i-run)                       ; 'true' if XEM file has run
      (progn
        (if xem-highlight-output-flag
            ;; non-occur
            (progn
              (kill-buffer "*Occur*")
              (delete-other-windows)
              (setq xem-highlight-output-flag nil)
              (message "  xem: toggle-highlight-output revert complete."))
          ;; occur
```

```
          (progn
            (xem-highlight-output)
            (setq xem-highlight-output-flag t)
            (message "  xem: toggle-highlight-output occur complete.")))
    (ding)                                  ; acknowledge visual bell setting
    (message "  xem: toggle-highlight-output: this XEM file not identified as having run.")))

;; ---------------------------------
;;  squish display
;; ---------------------------------

(defvar xem-squish-flag 0
  "Flag for `xem-cycle-squish', 0 = no squish, 2 = maximum squish.")
(make-local-variable 'xem-squish-flag)

(defun xem-cycle-squish ()
  "Cycle thru two levels of selective display.  The cursor color will also change."
  (interactive)
  (let ((level-zero  0 )
        (level-one   6 )
        (level-two   2 ))
    (cond ((= xem-squish-flag 0)
           (xem-selective-set level-one)     ; squish to depth 6
           (setq xem-squish-flag 1)
           (message "  xem: cycle-squish selective display set at indent %d." level-one))
          ((= xem-squish-flag 1)
           (xem-selective-set level-two)     ; squish to depth 2
           (recenter)
           (setq xem-squish-flag 2)
           (message "  xem: cycle-squish selective display set at indent %d." level-two))
          ((= xem-squish-flag 2)
           (xem-selective-set level-zero)             ; unsquish
           (setq xem-squish-flag 0)
           (message "  xem: cycle-squish selective display unset."))
          (t (message "  xem: cycle-squish: problem with xem-squish-flag %d" xem-squish-flag)))))

(defun xem-selective-set (indent)
  "Set `set-selective-display' to INDENT and change cursor color."
  (set-selective-display indent)
  (cond ((= indent 0) (set-cursor-color "black"))
        ((= indent 2) (set-cursor-color "cyan1"))
        ((= indent 6) (set-cursor-color "cyan3")))
  (setq xem-squish-flag t)
  (message "  xem: set-selective un/set using indent %d" indent))

;; ---------------------------------
;;  key/value functions
;; ---------------------------------

(defun xem-highlight-output ()
  "Show occurs for trigger 'r-highlight-output'."
  (let ((trigger "r-highlight-output"))
    (xem-key-occur (xem-value-get trigger))
    (message "  xem: highlight-output complete.")))

(defun xem-have-i-run ()
  "Examine 'used-svn' value and return t if substantial, else nil."
  (let ((trigger  "used-svn" )
        (value               ))
    (setq value (xem-value-get trigger))
    (message "  xem: have-i-run complete, recovered '%s'" value)
    (if (= (length value) 0) nil t)))        ; expose the return value

(defun xem-value-get (key &optional stripquotes)
  "Return value for input or output field key KEY and perhaps STRIPQUOTES.  Else return nil.

This function searches the current buffer for the given KEY.

Note that string values retain their double-quotes."
  (save-excursion                           ; hold cursor position
    (let ((regex  (concat "^[[:blank:]]+" key ".*[<>][[:blank:]]*\\(.*\\)$") )
          (value  nil                                                       ))
      (message "  xem: value-get: using search regex '%s'" regex)
      (goto-char (point-min))
      (if (re-search-forward regex
                             nil t)
          (setq value (match-string 1)))     ; 1 indicates first \( \) construct
      (if stripquotes
          (if (string-match "^\"\\(.*\\)\"$" value)
              (setq value (match-string 1 value))))
      (message "  xem: value-get: complete with key '%s' with recovered value '%s'" key value)
```

```
      value)))                                ; expose the return value

(defun xem-key-occur (key)
  "Run 'occur' on output fields containing KEY key.

KEY may be double-quoted, in which case the quotes are first removed."
  (save-excursion                             ; hold cursor position
    (let ((regex-1  "^\\\"\\(.*?\\)\\\"$" )   ; un-double-quote regex (note the triple escapes)
          (key-2    nil                   )
          (regex-2  nil                   ))
      (if (not key)
          (error "  xem: key-occur supplied nil key '%s'" key))
      (if (string-match regex-1 key)
          (setq key-2 (match-string 1 key)))   ; 1 indicates first \( \) construct
      (setq regex-2 (concat "^[[:blank:]]+" key-2 ".*<"))
      (message "  xem: key-occur: using occur regex '%s'" regex-2)
      (occur regex-2)
      (message "  xem: key-occur: complete"))))

;; ---------------------------------
;;  field navigation
;; ---------------------------------

(defun xem-is-field (wrangle)
  "Test current line for a particular field type determined by regex WRANGLE.

This function returns either t or nil.  WRANGLE is normally \">\"
for in-data, \"<\" for out-data, or \"<>\" for either."
  (let* ((regex  (concat "[" wrangle "]") )              ; add the regex list operator sytax
         (line   (thing-at-point 'line) ))
    (message "  xem: is-field: regex '%s'" regex)
    (if (and (not (string-match "<.*>\\|>.*<" line))     ; email addresses and similar are excluded
             (string-match regex line))
        t
      nil) ))

(defun xem-move-field-next ()
  "Move to next field."
  (interactive)
  (while (and (forward-line +1)
              (not (xem-is-field "<>"))
              (not (eobp))) )
  (end-of-line)
  (message "  xem: move next field: complete"))

(defun xem-move-field-prior ()
  "Move to previous field."
  (interactive)
  (while (and (forward-line -1)
              (not (xem-is-field "<>"))
              (not (bobp))) )
  (end-of-line)
  (message "  xem: move prior field: complete"))

(defun xem-move-field-in-next ()
  "Move to next in-data field."
  (interactive)
  (while (and (forward-line +1)
              (not (xem-is-field ">"))
              (not (eobp))) )
  (end-of-line)
  (message "  xem: move next in-data field: complete"))

(defun xem-move-field-in-prior ()
  "Move to previous in-data field."
  (interactive)
  (while (and (forward-line -1)
              (not (xem-is-field ">"))
              (not (bobp))) )
  (end-of-line)
  (message "  xem: move prior in-data field: complete"))

(defun xem-move-field-out-next ()
  "Move to next out-daat field."
  (interactive)
  (while (and (forward-line +1)
              (not (xem-is-field "<"))
              (not (eobp))) )
  (end-of-line)
  (message "  xem: move next out-data field: complete"))
```

```
(defun xem-move-field-out-prior ()
  "Move to previous out-data field."
  (interactive)
  (while (and (forward-line -1)
              (not (xem-is-field "<"))
              (not (bobp))) )
  (end-of-line)
  (message "  xem: move prior out-data field: complete"))

;; ---------------------------------
;;  entity navigation
;; ---------------------------------

(defconst xem-entity-move-regex (concat "^entity\\.\\|^" xem-dchar "[[:blank:]]*entity\\."))
(defconst xem-recenter 1) ; useful values: 0, 1, nil (to center)

(defun xem-move-entity-next ()
  "Navigate to next entity record."
  (interactive)
  (let ((regex  xem-entity-move-regex ))
    (if (looking-at regex)
        (forward-char 1))                    ; need to "kick-start" the search
    (if (re-search-forward regex
                           nil t)            ; limit (no), no error (true)
        (progn
          (beginning-of-line)
          (if (= xem-squish-flag 2)          ; squish-level sensitive 'recenter'
              () ;(recenter nil)
            (recenter xem-recenter))
          (message "  xem: move to next entity complete."))
      (beginning-of-line)
      (message "  xem: move to next entity FAILED."))
    (match-beginning 0) ))                   ; nil if no match

(defun xem-move-entity-prior ()
  "Navigate to prior (meaning previous) entity record."
  (interactive)
  (let ((regex  xem-entity-move-regex ))
    (if (re-search-backward regex
                            nil t)           ; limit (no), no error (true)
        (progn
          (beginning-of-line)                ; superfluous in this case
          (if (= xem-squish-flag 2)          ; squish-level sensitive 'recenter'
              () ; (recenter nil)
            (recenter xem-recenter))
          (message "  xem: move to prior entity complete."))
      (beginning-of-line)
      (message "  xem: move to prior entity FAILED."))
    (match-beginning 0) ))                   ; nil if no match

;; ---------------------------------
;;  hash navigation
;; ---------------------------------

(defconst xem-hash-move-regex (concat "^[[:blank:]]+" xem-dchar "[[:blank:]]*[[:alnum:]]"))
(defconst xem-hash-recenter nil) ; useful values: 0, 1, nil (to center)

(defun xem-move-hash-ring ()
  "Cycle to next disabled field, returning to the top when necessary."
  (interactive)
  (let ((here  (point) ))
    (if (xem-move-hash-next)
        ()
      (beginning-of-buffer)
      (if (xem-move-hash-next)
          ()
        (goto-char here)))))

(defun xem-move-hash-next ()
  "Navigate to next disabled field."
  (interactive)
  (let ((regex  xem-hash-move-regex ))
    (if (looking-at regex)
        (forward-char 1))                    ; need to "kick-start" the search
    (if (re-search-forward regex
                           nil t)            ; limit (no), no error (true)
        (progn
          (beginning-of-line)
          (recenter xem-hash-recenter)
          (message "  xem: move to next hash complete.")
          t)
```

```
      (beginning-of-line)
      (message "  xem: move to next hash FAILED.")
      nil)))

(defun xem-move-hash-prior ()
  "Navigate to prior (meaning previous) disabled field."
  (interactive)
  (let ((regex  xem-hash-move-regex ))
    (if (re-search-backward regex
                            nil t)           ; limit (no), no error (true)
        (progn
          (beginning-of-line)                ; superfluous in this case
          (recenter xem-hash-recenter)
          (message "  xem: move to prior hash complete."))
      (beginning-of-line)
      (message "  xem: move to prior hash FAILED."))))

;; ---------------------------------
;;  narrowing
;; ---------------------------------

(defvar xem-narrow-flag 0
  "Flag for `xem-toggle-narrow', 0 = wide, 1 = narrow.")
(make-local-variable 'xem-narrow-flag)

(defun xem-toggle-narrow ()
  "Toggle between `xem-narrow-to-entity' and `xem-widen'."
  (interactive)
  (cond ((= xem-narrow-flag 0)
         (xem-narrow-to-entity))
        ((= xem-narrow-flag 1)
         (xem-widen))))

;; assumes (put 'narrow-to-region 'disabled nil)

(defconst xem-note-move-regex "^note$")           ; notes cannot be disabled
(defconst xem-rule-regex "^[[:blank:]]+------")   ; can add second blank for a more restricted view

(defun xem-widen ()
  "Widen buffer."
  (interactive)
  (save-excursion
    (widen)
    (recenter xem-recenter)
    (customize-set-variable 'cursor-type 'box)    ; default cursor
    (setq xem-narrow-flag 0)
    (message "  xem: widen complete.") ))

(defun xem-narrow-to-entity ()
  "Narrow to current entity.  Use \\[widen] to widen.

Requires the cursor be at the start of the entity in question.

Will need the user to set (put 'narrow-to-region 'disabled nil)."
  (interactive)
  (save-excursion
    (let ((regex  xem-note-move-regex )
          (beg    (point-min)         )
          (end    (point-max)         )
          (tmp    (point-max)         ))
      (if (looking-at xem-entity-move-regex)
          (progn
            (setq beg (point))
            (xem-move-entity-next)
            (setq end (point))
            ;; deal with last entity
            (if (= beg end)
                (if (re-search-forward regex
                                       nil t)
                    (setq end (match-beginning 0))))
            ;; deal with rules
            (goto-char beg)
            (if (re-search-forward xem-rule-regex
                                   nil t)
                (setq tmp (match-beginning 0)))
            (if (< tmp end)
                (setq end tmp))
            (if (not (= beg end))            ; protect against second narrowing
                (progn
                  (narrow-to-region beg end) ; opposite is (widen)
                  (setq xem-narrow-flag 1)
```

```
                (customize-set-variable 'cursor-type 'hollow)
                (customize-set-variable 'cursor-type 'bar)
                (message "  xem: narrow to entity complete, %d to %d." beg end))
            (message "  xem: narrow to entity already applied.")))
        (message "  xem: narrow to entity requires cursor be positioned at start of target entity."))
      )))

;; ---------------------------------
;;  colorizing
;; ---------------------------------

;; Xll colors  : /etc/X11/rgb.txt
;; show colors : file:///home/robbie/synk/xeona/x11-colors/x11-color-names.html
;; package     : /usr/share/emacs/21.4/lisp/hi-lock.el

;; interesting: aquamarine gold salmon yellow lemon_chiffon

;; complain if not Xll
(if (eq window-system 'x)                    ; nil = ordinary terminal, 'w32 = MSWin
    (message "  xem: X11 found.")
  (error "  xem: WARNING: X11 assumed but not found"))

;; add support package
(require 'hi-lock)

;; define colors
(defface xem-hi-color-B01
  '((((background dark)) (:background "wheat" :foreground "black"))
    (t (:background "wheat")))              ; this entry is normally used
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B02
  '((((background dark)) (:background "gold" :foreground "black"))
    (t (:background "gold")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B03
  '((((background dark)) (:background "wheat" :foreground "black"))
    (t (:background "wheat")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B04
  '((((background dark)) (:background "salmon" :foreground "black"))
    (t (:background "salmon")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B05
  '((((background dark)) (:background "gold" :foreground "black"))
    (t (:background "gold")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B06
  '((((background dark)) (:background "burlywood" :foreground "black"))
    (t (:background "burlywood")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B07
  '((((background dark)) (:background "beige" :foreground "black"))
    (t (:background "beige")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-B08
  '((((background dark)) (:background "sandy brown" :foreground "black"))
    (t (:background "sandy brown")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F01
  '((t (:weight normal :foreground "steel blue")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F02
  '((t (:weight normal :foreground "deep pink")))
```

```
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F03
  '((t (:weight normal :foreground "peru")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F04
  '((t (:weight normal :foreground "tomato")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F05
  '((t (:weight normal :foreground "tan")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F06
  '((t (:weight normal :foreground "black")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xem-hi-color-F07
  '((t (:weight normal :foreground "peru")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

;; define colorizing regexes
(defconst xem-entity-tmp-regex  "^entity\\..*-0$")
(defconst xem-entity-regex      "^entity\\.")
(defconst xem-entity-dis-regex  (concat "^" xem-dchar "[[:blank:]]*entity\\."))
(defconst xem-record-regex      "^program\\.\\|^model-end$")
(defconst xem-note-regex        "^note")
(defconst xem-class-regex       "class[[:blank:]]+>.*$")
(defconst xem-strong-regex      "^    [[:blank:]]*\\*\\*+[[:blank:]][[:print:]]+$")
(defconst xem-field-dis-regex   (concat xem-dchar "[[:blank:]][[:graph:]][[:print:]]+$"))
(defconst xem-string-regex      "\".*?\"")
(defconst xem-comment-regex     "^      [[:blank:]]*[[:graph:]]")
(defconst xem-header-regex      "header: [[:graph:]]+")

;; define overarching function
(defun xem-colorize ()
  "Colorize a XEM file."
  (interactive)
  ;; CAUTION: order is important
  (highlight-lines-matching-regexp xem-entity-tmp-regex 'xem-hi-color-B08)
  (highlight-lines-matching-regexp xem-entity-regex     'xem-hi-color-B07)
  (highlight-lines-matching-regexp xem-entity-dis-regex 'xem-hi-color-B04)
  (highlight-lines-matching-regexp xem-record-regex     'xem-hi-color-B01)
  (highlight-lines-matching-regexp xem-note-regex       'xem-hi-color-B06)
  (highlight-regexp                xem-class-regex      'xem-hi-color-B02)
  (highlight-regexp                xem-strong-regex     'xem-hi-color-F02)
  (highlight-regexp                xem-field-dis-regex  'xem-hi-color-F03)
  (highlight-regexp                xem-string-regex     'xem-hi-color-F04)
  (highlight-lines-matching-regexp xem-comment-regex    'xem-hi-color-F05)
  (highlight-regexp                xem-header-regex     'xem-hi-color-F07)
  (message "  xem: colorize: colorizing complete"))

;; show colors : firefox file:///home/robbie/synk/xeona/x11-colors/x11-color-names.html
;; show colors : `list-colors-display'

;; ---------------------------------
;;  toggle data
;; ---------------------------------

;; make records and fields active and inactive

(defun xem-toggle-data ()
  "Toggle enable on records and fields.

Caution: there needs to be one uppermost ACTIVE field."
  (interactive)
  (let ((dchar  xem-dchar )
        (line   nil       ))
    (setq line (thing-at-point 'line))
    (message "  xem: toggle-data: line '%s'" line)

    (defun replace (nline)                              ; assumes 'nline' has trailing newline
      (beginning-of-line)
      (kill-line nil)
```

```
      (insert nline))

    (cond                                               ; elisp case statement
     ((string-match "^program\\.[[:alnum:]]"   line)    ; record identifier "entity.[a-z..].."
      (message "  xem: toggle-data record complete (no action taken)."))
     ((string-match "^entity\\.[[:alnum:]]"    line)
      (replace (concat dchar " " line))
      (forward-line -1)
      (message "  xem: toggle-data record complete."))
     ((string-match (concat dchar "[[:blank:]]*entity\\.[[:alnum:]]")  line)
      (replace (substring line 2 nil))                  ; toggle disabled, now explicit leading "# "
      (forward-line -1)
      (message "  xem: toggle-data record complete."))
     ((string-match "^[[:blank:]]*[[:alnum:]]" line)
      ;; could shift some code to 'xem-activate-field'
      (end-of-line)
      (insert "\n")
      (replace (xem-toggle-disable line dchar))
      (forward-line -1)
      (message "  xem: toggle-data disable field complete."))
     (t
      (xem-activate-field)) )))                         ; toggle disabled

(defun xem-activate-field ()
  "Support for `xem-toggle-data'."
  (let ((dchar  xem-dchar )                 ; xem disable character (assumed length one)
        (king   nil       )                 ; xem field line to be promoted
        (count  0         ))                ; shuffle count
    ;; local function
    (defun replace (nline)
      (beginning-of-line)
      (kill-line nil)
      (insert nline))
    ;; main code
    (if (xem-is-enabled (thing-at-point 'line) dchar)
        (message "  xem: activate-field: nothing to do.")  ; final message
      ; else block
      (setq king (thing-at-point 'line))
      (setq king (xem-toggle-disable king dchar))
      (replace "")                          ; effectively kill entire line
      (forward-line -1)                     ; move up
      (setq count 1)
      ;; loop
      (while (not (xem-is-enabled (thing-at-point 'line) dchar))
        (forward-line -1)
        (incf count))
      ;; completion
      (replace (xem-toggle-disable (thing-at-point 'line) dchar))
      (forward-line -1)
      (insert king)
      (message "  xem: activate-field complete with %d shuffles up." count)))) ; final message

(defun xem-toggle-disable (line dchar)
  "For supplied field LINE string, toggle the disable DCHAR."
  ;; assumes "  # " style alignment
  (if (xem-is-enabled line dchar)
      (concat "  "  dchar  " " (substring line 4 nil))
    (concat "    " (substring line 4 nil))))

(defun xem-is-enabled (line dchar)
  "Report as to whether supplied field LINE string is DCHAR disabled."
  (let ((first (car (split-string line))))   ; use default separator "[ \f\t\n\r\v]+"
    (not (equal dchar first))))              ; strings compared element by element

;; ---------------------------------
;;  occur calls
;; ---------------------------------

(defun xem-occur-builtin-remarks ()
  "List builtin-remark lines."
  (interactive)
  (let ((regex  "builtin-remark" ))
    (occur regex 0)
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-builtin-remarks complete, regex '%s'." regex)))

(defun xem-occur-entities ()
  "List entity lines."
  (interactive)
  (let ((regex  "^entity\\." ))
    (occur regex 0)
```

```
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-entities complete, regex '%s'." regex)))

(defun xem-occur-entities-disabled ()
  "List disabled entities."
  (interactive)
  (let ((regex  (concat "^" xem-dchar) ))
    (occur regex 0)
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-entities-disabled complete, regex '%s'." regex)))

(defun xem-occur-classes ()
  "List class lines."
  (interactive)
  (let ((regex  "class[[:space:]]*>" ))
    (occur regex 0)
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-classes complete, regex '%s'." regex)))

(defun xem-occur-rules ()
  "List rules."
  (interactive)
  (let ((regex  "^[[:blank:]]+---+ [[:alnum:] ]+$" ))  ; at least three dashes are required "---"
    (occur regex 0)
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-rules complete, regex '%s'." regex)))

(defun xem-occur-hashes ()
  "List disabled field lines."
  (interactive)
  (let ((regex  (concat "^[[:blank:]]+" xem-dchar) ))
    (occur regex 2)
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-hashes complete, regex '%s'." regex)))

(defun xem-occur-identities ()
  "Show {actor,asset,block,commodity,..} identity lines."
  (interactive)
  (let ((enlist   nil )                     ; user-defined list of entities phrases
        (entity   nil )
        (entitys  nil )                     ; 'enlist' in \\|-separated form
        (iflist   nil )                     ; user-defined list of interface phrases
        (iface    nil )
        (ifaces   nil )
        (regex    nil ))                    ; 'occur' regex
    ;; add to these as required
    (setq enlist "actor asset block commodity context domain junction operator selgate")
    (setq enlist (concat enlist " " "[1-9]")); for "socket-1" and so on
    (setq iflist "[[:alnum:]-]+")           ; earlier also "cable socket"
    ;; active code
    (dolist (entity (split-string enlist))   ; use default separator "[ \f\t\n\r\v]+"
      (setq entitys (concat entitys entity "\\|")))
    (dolist (iface (split-string iflist))    ; use default separator "[ \f\t\n\r\v]+"
      (setq ifaces (concat ifaces iface "\\|")))
    (setq entitys (substring entitys 0 -2))  ; trim the final "\\|", -2 is correct
    (setq ifaces  (substring ifaces  0 -2))  ; trim the final "\\|", -2 is correct
    (setq regex (concat "\\(" ifaces "\\)-\\(" entitys "\\)s?[[:space:]]+[lL]"))
    (occur regex)
    (shrink-window-if-larger-than-buffer (next-window))
    (message "  xem: occur-identities complete, regex '%s'." regex)))

;; ---------------------------------
;;  insert entity calls
;; ---------------------------------

(defun xem-insert-entity-regex (&optional xeona)
  "Attempt to insert sought XEDOCs into current buffer, optionally using XEONA.

The user will be prompted for a class name regex."
  (interactive)
  (let ((option  "--class" )                ; xeona option
        (class   ""        )                ; entity class name
        (xedoc   ""        ))               ; sought XEDOC or XEDOCs
    (if (not xeona)                         ; xeona name not set externally
        (setq  xeona xem-binary))

    (if (file-executable-p xeona)           ; xeona is executable by you
        ;; main code
        (progn
          (setq class
                (read-string
```

```
                "Enter case-sensitive class name regex [default .]: "     ; prompt
                nil                                                       ; initial
                nil                                                       ; history list
                "."))                                                     ; set default

          (setq call (concat xeona " " option " " class " " "2>/dev/null"))
          (setq xedoc (shell-command-to-string call))

          (if (not (looking-at "^")) (progn (end-of-line) (insert "\n")))
          (insert xedoc)
          ;; completion reporting
          (message "  xem: insert-entity-regex complete using class regex '%s'." class))

      (progn
        ;; executable not found reporting
        (message "  xem: insert-entity-regex FAILURE, xeona executable not found '%s'." xeona)))))

(defun xem-insert-entity-scroll ()
  "Select an entity from a refreshed list of entities."
  (interactive)
  (let ((binary   xem-binary                                                )
        (args1   "--class + 2>/dev/null | gawk '/[[:alpha:]]/ { print $1 }'" )   ; [1]
        (args2   "--class"                                                  )
        (args3   "2>/dev/null"                                              )
        (call                                                               )
        (entity-list                                                        )
        (class                                                              ))
    ;; [1] the "/[[:alpha:]]/" is to screen out blank lines
    ;; check for 'gawk' utility
    (if (equal "" (shell-command-to-string "which gawk"))
        (message "  xem: insert-entity-scroll: 'gawk' utility not present on system"))
    ;; active code
    (if (file-executable-p binary)           ; 'xeona' is executable by you
        ;; executable found
        (progn
          (setq call (concat binary " " args1))
          (setq entities (shell-command-to-string call))
          (setq entity-list (split-string entities "\n"))
          (message "  xem: insert-entity-scroll: entity list length %d" (length entity-list))
          (setq class
                (read-string
                 "Scroll and enter class name: "  ; prompt
                 (pop entity-list)                ; initial
                 'entity-list                     ; history list
                 nil))                            ; set default

          (setq call (concat binary " " args2 " " class " " args3))
          (setq xedoc (shell-command-to-string call))

          (when (not (looking-at "^")) (end-of-line) (insert "\n"))
          (insert xedoc)
          ;; completion reporting
          (message "  xem: insert-entity-scroll complete using class regex '%s'." class))
      (progn
        ;; executable not found reporting
        (message "  xem: insert-entity-scroll FAILURE, xeona executable not found '%s'." binary)))))

;; ---------------------------------
;;  file alignment calls
;; ---------------------------------

(defun xem-reindent ()                       ; quite well coded, see RCS:1.538 for different 'fmtstr'
  "Reset a XEM file based on prevailing 'tab-stop-list' setting."
  (interactive)
  (save-excursion
    (let ((prior     nil        )  ; prior 'kill-whole-line' setting
          (indent-1  4          )  ; desired indent from existing 'tab-stop-list', else accept this
          (indent-2  55         )  ; desired indent from existing 'tab-stop-list', else accept this
          (fmtstr    nil        )  ; reformat string for 'format' call
          (tags      '(">" "<") )  ; indicating input and output data lines
          (line      nil        )  ; prevailing line
          (chunks    nil        )  ; split line
          (count     0          )) ; processed lines count
      ;; preamble
      (setq prior kill-whole-line)           ; keep for later
      (setq kill-whole-line nil)
      ;; set indents
      (goto-char (point-min))                    ; park point at beginning of buffer
      (if (search-forward "tab-stop-list" nil t)  ; 'nil' to end of buffer, 't' means no error on fail
          (progn
            (setq indent-1 (nth 0 tab-stop-list))      ; nth 0 is the first element!
```

```
            (setq indent-2 (nth 1 tab-stop-list))))    ; nth 1 is the second element!
      ;; main code
      (goto-char (point-min))                ; park point at beginning of buffer
      (while (not (eobp))
        (setq line (thing-at-point 'line))   ; contains trailing newline
        (dolist (tag tags)                   ; note shortened form with default return
          (when (string-match tag line)
            (setq chunks (split-string line tag))  ; returns list, also modifies the match data
            (setq fmtstr (format "%%-%ds %%s %%s" (1- indent-2)))  ; minus one for padding
            (message "  xem: fmtstr '%s'" fmtstr)
            (insert
             (format fmtstr (robbie-rdeblank (pop chunks)) tag (robbie-ldeblank(pop chunks))))
            (kill-line nil)              ; leave terminating newline if 'kill-whole-line' is 'nil'
            (delete-char -1)             ; back up too!
            (incf count) ))
        (forward-line +1) )                  ; creep forward
      ;; housekeeping
      (setq kill-whole-line prior)           ; reinstate previous setting
      (delete-trailing-whitespace)           ; remove trailing whitespace / can be commented out
      (message "  xem: reindent complete with %d lines processed using %d indent." count indent-2) ))
  (beginning-of-line))                       ; outside 'save-excursion' block

;; the following 'xem-reset-tabstops' is now mostly redundant
;; because 'xeona' automatically does this from r2299

(defun xem-reset-tabstops ()
  "Reset the prevailing 'tab-stop-list' based on current indentation."
  (interactive)
  (save-excursion
    (let ((lines    0           )            ; number of lines
          (tab      0           )            ; current second tab setting
          (index    0           )            ; angle index
          (line     ""          )            ; captured line
          (lo       0           )            ; lowest angle index
          (hi       0           )            ; highest angle index
          (singles  0           )            ; number of single angles { < > }
          (doubles  0           )            ; number of multiple angles, such as email addresses
          (indexes  ()          )            ; empty list
          (change   "no change" ))           ; completion reporting
      ;; get current tab value
      (setq tab (nth 1 tab-stop-list))
      (message "  xem: reset-tabstops: tab %d" tab)
      ;; loop thru lines
      (goto-char (point-min))                ; park point at beginning of buffer
      (while (not (eobp))
        (setq line (thing-at-point 'line))   ; contains trailing newline
        ;; examine line
        (if (and (string-match "<" line)
                 (string-match ">" line))    ; exclude "< >" style constructs
            (progn
              (incf doubles))
          (progn
            (if (or (string-match "<" line)  ; output mark
                    (string-match ">" line)) ; input mark
                (progn
                  (incf singles)
                  (setq index (1- (match-end 0)))
                  (push index indexes) ))))
        ;; creep forward
        (incf lines)
        (forward-line +1) )
      ;; modify tab-stop-list given indexes are consistent
      (when indexes
        (setq indexes (sort indexes '<)) ; CAUTION: must reassign 'indexes'
        (setq lo (nth 0 indexes))
        (setq hi (car (last indexes)))
        (message "  xem: reset-tabstops: lo %d hi %d" lo hi)
;;;     (if (and (= lo hi) (not (= lo tab)))       ; old code
        (if (or (not (= lo hi)) (not (= lo tab)))  ; new code
            (progn
              (goto-char (point-min))
              (setq occurrence (re-search-forward "tab-stop-list: " nil t))
              (if occurrence
                  (progn
                    (setq change "modified")
                    (goto-char occurrence)
                    (kill-line)          ; delete to end of line
;;;                     (setq tabstops (format "(%02d %02d %02d)" 4 lo (+ lo 2)))    ; old code
                    (setq tabstops (format "(%02d %02d %02d)" 4 hi (+ hi 2)))    ; new code
                    (insert tabstops) )))))
      ;; completion reporting
```

```
      (setq msg1 (format "reset-tabstops complete: %s." change))
      (setq msg2 (format "hits: %d lines, %d singles, %d doubles (email)." lines singles doubles))
      (setq msg3 (format "second tab: old = %d, lo = %d, hi = %d / tabstops = %s." tab lo hi tabstops))
      (message "  xem: %s\n%s\n%s." msg1 msg2 msg3))))

;; ---------------------------------
;;  rule calls
;; ---------------------------------

(defun xem-rule (&optional indent)
  "Create standard separator rules using optional INDENT.

The separator rules take the form:

  ---------------------------------------- mandatory entities

If necessary, the user will be prompted for text."
  (interactive)
  (let ((default-indent xem-rule-indent )    ; indent
        (overall        60              )    ; final line length
        (line           nil             )
        (words          nil             )
        (text           nil             )
        (len            nil             ))
    (if indent () (setq indent default-indent))
    ;; function 'split-string': default 'SEPARATOR' is
    ;; "[ \f\t\n\r\v]+", empty matches do count when not
    ;; adjacent to another match
    (setq line (thing-at-point 'line))       ; contains trailing newline
    (setq words (split-string line))
    ;; establish 'text'
    (if (equal words nil)                    ; 'nil' if only non-words (meaning spaces)
        (progn
          (setq text
                (read-string
                 "Enter text [default 'xxx']: "   ; prompt
                 nil                              ; initial
                 nil                              ; history list
                 "xxx")))                         ; set default
      (progn
        (if (string-match "-+" (car words))       ; '+' is match-one-or-more operator
            (pop words))
        (while words
          (setq text (concat text (pop words) " ")))
        (setq text (substring text 0 -1)) ))  ; remove final blank
    ;; remake line
    (setq len (- overall (+ indent (length text) 1)))
    (setq output (concat  (make-string indent 32)   (make-string len ?-) " " text))
    ;; replace current line
    (beginning-of-line)
    (kill-line)
    (insert output)
    (insert "\n")
    (forward-char -1)                        ; realign point
    ;; completion reporting
    (message "  xem: rule complete using text '%s'." text)))

;; ---------------------------------
;;  scan-builtin-remark
;; ---------------------------------

(defun xem-extract-data (line                ; input (no newline)
                         key                 ; sought key
                         splif)              ; split string regex, usually " < " or " > "
  "Extract field data from the given LINE, KEY, and SPLIF regex."
  (let ((data     nil )                      ; return value
        (element  nil )
        (first    nil )
        (second   nil )
        (temp     nil )
        (key2     nil ))
    (setq element (split-string line splif))
    (when (< 0 (length element))
      (setq first  (xem-string-trim (pop element)))
      (setq second (xem-string-trim (pop element)))
      (setq temp   (split-string first))
      (setq key2   (xem-string-trim (pop temp)))
      (if (string-match key key2)
          (setq data second)))
    (if data (message "  xem: extract-data : %s %-30s %s" splif key data))
    data ))                                  ; expose the return value
```

```
(defun xem-unique (list)
  "Remove adjacent duplicate elements from the supplied LIST using `equal'.

It is recommended the list be sorted first, for instance:

    (setq results (sort results 'string<))  ; sort lexicographically"
  (let ((delay       "" )
        (element        )
        (duplicates  0  )
        (output         ))
    (if (not (listp list))                  ; list, includes 'nil'
        (message " xem unique  : list failed predicate test")
      (dolist (element list)                ; else
        (if (not (equal element delay))
            (push element output)
          (setq duplicates (+ 1 duplicates)))
        (setq delay element))
      (setq output (nreverse output))       ; reverse the order (can also use 'reverse')
      (message "  xem: unique  : complete, excluded duplicates %d" duplicates)
      output)))                             ; expose the return value

(defun xem-scan-class-buffer ()
  "Scan buffer and extract and print sorted 'class' and 'builtin-remark' data."
  (interactive)
  (let ((lines   0               )          ; number of lines processed
        (buffer  (current-buffer) ))         ; current buffer object
    (setq lines (xem-scan-class-region (point-min) (point-max)))
    (split-window-vertically)
    (switch-to-buffer buffer)
    (message "  xem: scan-class-buffer complete, %d lines processed." lines)))

(defun xem-scan-class-region (reg-start reg-end)
  "Scan REG-START and REG-END region and extract and print sorted 'class' and 'builtin-remark' data."
  (interactive "r")
  (save-excursion
    (let ((buffer-name  "*XEM-builtin-remarks*" )      ; results buffer
          (result       nil                     )      ; formatted result
          (results      nil                     )      ; vector of 'result'
          (linebuf      nil                     )      ; line buffer
          (class        nil                     )      ; class data
          (remark       nil                     )      ; builtin-remark data
          (lines        0                       ))     ; lines processed
      ;; grab data
      (goto-char reg-start)                       ; goto start region
      (while (< (point) reg-end)                  ; cycle thru lines in the region
        (setq linebuf (thing-at-point 'line))     ; contains trailing newline
        (if (string-match "\n$" linebuf)          ; remove trailing newline
            (setq linebuf (substring linebuf 0 (match-beginning 0)))) ; zero is string beginning
        (if (not class)  (setq class  (xem-extract-data linebuf "class"          " > ")))
        (if (not remark) (setq remark (xem-extract-data linebuf "builtin-remark" " < ")))
        (while remark
          (while (string-match "\"" remark)   ; CAUTION: 'while' not 'if' necessary
            (setq remark (replace-match "" t t remark)))
          (if (string-match "^$" remark) (setq remark "(not set)"))
          (push (format "    %-25s %s" class remark) results)
          (setq class  nil)
          (setq remark nil))
        (setq lines (+ lines 1))
        (forward-line 1))
      ;; sort output
      (setq results (sort results 'string<))      ; sort lexicographically
      (setq results (xem-unique results))         ; local function
      ;; (setq results (nreverse results))         ; reverse the order (can also use 'reverse')
      ;; new buffer
      (switch-to-buffer buffer-name)
      (text-mode)
      (setq truncate-lines t)
      ;; insert results
      (insert (format "unique sorted classes: %d\n" (length results)))
      (dolist (result results)                    ; cycle thru lines, non-intrusive
        (insert result)
        (insert "\n"))
      ;; housekeeping
      (goto-char (point-min))                     ; go back to the beginning
      (message "  xem: scan-class-region complete, %d lines processed." lines)
      lines)))                                    ; expose the return value

;; ---------------------------------
;;  insert rule
;; ---------------------------------
```

```
(defun xem-insert-rule-scroll ()
  "Insert rule using standard list."
  (interactive)
  (let ((rule-string)
        (rules (list "program admin"
                     "mandatory entities"
                     "domain 1"
                     "operators"
                     "entities"
                     "nodes"
                     "commodities"
                     "contexts"
                     "tail")))
    ;; query user
    (setq rule-string
          (read-string
           "Scan rule strings: "             ; prompt
           (pop rules)                       ; initial
           'rules                            ; history list
           ""))                              ; set default
    ;; insert rule
    (while (not (looking-at "^$")) (end-of-line) (insert "\n"))
    (insert rule-string)
    (insert "\n")
    (forward-line -1)
    (xem-rule)                         ; process rule
    (message "  xem: insert-rule complete using '%s'." rule-string)))

;; ---------------------------------
;;  provide
;; ---------------------------------

;; put the mode symbol into the list "features", so that users can
;; invoke (require 'xem-mode) and load your code only when needed

(provide 'xem-mode)

;;; xem.el ends here

;  $Id: xem.el 5309 2010-10-13 08:03:46Z robbie $
;  end of file
```

```
;;; xem-1.el --- additional code 1 for 'xem.el'

;  file-purpose     : emacs additional code 1 for 'xem.el'
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Tue 24-Aug-2010 09:51 UTC
;  file-status      : ongoing
;  file-keywords    : emacs xeona

;  $Revision: 5181 $
;  $Date: 2010-08-24 15:07:53 +0200 (Tue, 24 Aug 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xem-1.el $

;; ---------------------------------
;;  rename utilities
;; ---------------------------------

(defun xem-uncamel ()
  "Uncamel current symbol using `xem-decamel'."
  (interactive)
  (let ((beg       nil )                          ; symbol start
        (end       nil )                          ; symbol end
        (original  nil ))                         ; current identifier
    ;; grab the current identifier
    (setq original (thing-at-point 'symbol))      ; normal word plus '_' for C language
    (if (stringp original)                        ; check for meaningful string
        (progn
          ;; move to start of the identifier (decimal character codes are used)
          (if (not (or (= (char-before) 10)       ; peek backwards and check for newline
                       (= (char-before) 32)))     ; peek backwards and check for space
              (backward-word 1))
          ;; process
          (setq beg (point))
          (forward-char (length original))
          (setq end (point))
          (kill-region beg end)
          (insert (xem-decamel original "-"))
          ;; completion reporting
          (message "  xem: uncamel complete using '%s'." original))
      (message "  xem: uncamel, nothing to process."))))

(defun xem-decamel (symbol &optional replace)
  "Decamel SYMBOL using underscores or optionally the string REPLACE.
For example (note the behavior of multiple uppercasing):\n
    someVarName -> some_var_name
    abCD_fg     -> ab-cd_fg\n"
  (let ((case-fold-search  nil )                  ; CAUTION: must make case sensitive
        (default-replace  "_"  )                  ; default to underscore
        (output            nil ))
    (if (not replace) (setq replace default-replace))
    (while (string-match "\\([[:lower:]]\\|[[:digit:]]\\)\\([[:upper:]]\\)" symbol)
      (setq symbol
            (replace-match (concat "\\1" replace "\\2") t nil symbol)))    ; fixed case, literal (no)
    (downcase symbol)))

;; ---------------------------------
;;  general utilities
;; ---------------------------------

(defun xem-string-trim (string)
  "Trim STRING from both ends.

Unlike the standard version, this function returns an empty string if supplied a nil."
  ;; [:space:] comprises space, carriage-return, newline, tab, vertical-tab, form-feed
  (let ((regex  "^[[:space:]]*\\(.*?\\)[[:space:]]*$" ))    ; CAUTION: non-greedy '*?' required
    (if (not (stringp string))
        ""                                    ; else could be () to return 'nil'
      (if (string-match regex string)
          (match-string 1 string)             ; 1 indicates first \( \) construct
        string))))                            ; no match, return original

(defun xem-squeeze-lines ()
  "Automatically squeeze all multiple strictly-blank lines in current buffer."
  (interactive)
  (save-excursion                             ; hold cursor position
    (let ((sections     0   )                 ; sections treated
          (lines        0   )                 ; lines processed
          (sect-plural  "s" )                 ; "section" pluralization
          (line-plural  "s" ))                ; "line" pluralization
      (setq lines (count-lines (point-min) (point-max)))              ; original line count
      (goto-char (point-min))      ; start at beginning
```

```
      (while (re-search-forward    ; look for matches
              "\n\n\n+"
              nil t)               ; optional buffer position bound; return 'nil' not error on fail
        (replace-match
         "\n\n"
         t t)                      ; do not alter case; insert literally
        (setq sections (+ 1 sections)))
      (goto-char (point-min))      ; start at beginning again
      (while (looking-at "\n\n")   ; look for special match
        (kill-line)
        (setq sections (+ 1 sections)))
      (setq lines (- lines (count-lines (point-min) (point-max))))    ; original minus final line count
      (if (= sections 1) (setq sect-plural ""))
      (if (= lines    1) (setq line-plural ""))
      (message "  xem: squeeze-lines complete, processed %d section%s totaling %d line%s."
               sections sect-plural lines line-plural)) ))

(defun xem-trim-trailing ()
  "Trim trailing white space."
  (interactive)
  (delete-trailing-whitespace)
  (message "  xem: trim trailing whitespace complete"))

(defun xem-tar-interpret (code)
  "Interprete GNU tar exit CODE."
  ;; CAUTION: no 'message' calls so that this function can be
  ;; used directly in completion reports
  (let* ((msg  nil))
    (unless (integerp code)
      (error "  xem: tar-interpret: error: non-integer argument given"))
    (cond
     ((= code 0) (setq msg (format "%d = success"                                    code)))
     ((= code 1) (setq msg (format "%d = some files differ (certain options only)" code)))
     ((= code 2) (setq msg (format "%d = unspecified fatal errors"                   code)))
     (t          (setq msg (format "%d = undocumented return"                        code))))
    msg ))                                    ; expose return value

;  $Id: xem-1.el 5181 2010-08-24 13:07:53Z robbie $
;  end of file
```

```
;;; xem-2.el --- additional code 2 for 'xem.el'

;  file-purpose     : emacs additional code 2 for 'xem.el'
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Tue 24-Aug-2010 09:52 UTC
;  file-status      : ongoing
;  file-keywords    : emacs xeona

;  $Revision: 5181 $
;  $Date: 2010-08-24 15:07:53 +0200 (Tue, 24 Aug 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xem-2.el $

;; ---------------------------------
;;  dump to tar
;; ---------------------------------

(defconst xem-buffer-dump-tar-capture
  "*xem-dump-tar-capture*"
  "Buffer for tar output.")

(defconst xem-buffer-dump-readme-text
  "*xem-dump-readme-text*"
  "Buffer for readme file post-view.")

(defvar xem-dump-get-tag-history
  nil
  "Get history for `xem-dump'.")

(defun xem-dump-get-tag ()
  "Get tag string for `xem-dump'."
  ;; entered spaces get converted to "-"
  (let* ((prompt  "Enter a tarfile tag (may omit hyphen, can be null, spaces okay): " )
         (tag (read-string
               prompt                         ; prompt
               nil                            ; initial
               'xem-dump-get-tag-history      ; history list
               "") ))                         ; set default
    (setq tag (xem-string-trim tag))                              ; trim leading and trailing spaces
    (setq tag (replace-regexp-in-string "[[:blank:]]+" "-" tag)) ; replace remaining blanks
    (unless (or (string-match "^$" tag)       ; empty string
                (string-match "^-" tag))      ; hyphen offered
      (setq tag (concat "-" tag)))
    (message "  xem: dump-get-role: obtained tag '%s'" tag)
    tag ))                                    ; expose either empty or hyphen-lead string

(defvar xem-dump-get-role-history
  nil
  "Role history for `xem-dump'.")

(defun xem-dump-get-role ()
  "Get role string for `xem-dump'."
  (let* ((default  "(not given)"                                  )
         (prompt  (concat "Enter a tar role [" default "] : ") )
         (role (read-string
                prompt                         ; prompt
                nil                            ; initial
                'xem-dump-get-role-history     ; history list
                default) ))                    ; set default
    (message "  xem: dump-get-tag: obtained role '%s'" role)
    role ))                                   ; expose non-empty string

;; based on `minibuffer-local-map'

(defvar xem-dump-get-readme-keymap
  '(keymap
    (menu-bar
     keymap
     (minibuf
      "Xdump"
      keymap
      (previous menu-item "Previous readme" previous-history-element :help "Previous readme")
      (next menu-item "Next readme" next-history-element :help "Next readme")
      (return menu-item "Finish" exit-minibuffer :help "Terminate input and exit minibuffer")
      (quit menu-item "Quit" abort-recursive-edit :help "Abort input and exit minibuffer")
                    "Xdump"))
    (10 . exit-minibuffer)                    ; ^J
;;;;(13 . exit-minibuffer)                    ; ^M or <RET>, disable because we want natural editing
    ( 7 . abort-recursive-edit)               ; ^G
    (C-tab . file-cache-minibuffer-complete)
    ( 9 . self-insert-command)
```

```
    (up . previous-history-element)
    (prior . previous-history-element)
    (down . next-history-element)
    (next . next-history-element)
    (27 keymap
        (114 . previous-matching-history-element)
        (115 . next-matching-history-element)
        (112 . previous-history-element)
        (110 . next-history-element)))
  "Custom minibuffer keymap for `xem-dump'.")

(defvar xem-dump-get-readme-history
  nil
  "Readme input history for `xem-dump'.")

(defun xem-dump-get-readme (readfilename)
  "Seek optional readme text and stream to READFILENAME file.

This function provides interface support for function `xem-dump'."
  (let* ((prompt-1  "Add additional readme text? "         )
         (prompt-2  "Enter readme text (C-j to finish):\n" )
         (readme    nil                                   )
         (code      nil                                   )
         (chars     0                                     ))
    (if (y-or-n-p prompt-1)
        (progn
          (setq readme
                (read-from-minibuffer
                 prompt-2                          ; prompt-string: defined above
                 nil                               ; initial-contents: note last history via M-p
                 xem-dump-get-readme-keymap        ; keymap: 'nil' means use `minibuffer-local-map'
                 nil                               ; read: 'nil' means no convert text to lisp objects
                 'xem-dump-get-readme-history      ; hist: history object
                 t                                 ; inherit-input-method: 't' is inherit current state
                 ))
          (message "  xem: dump-get-readme: string:")
          (message "%s" readme)
          (setq chars (length readme))
          (setq ecall (concat "echo \"" readme "\" >> " readfilename)) ; CAUTION: soft-quotes essential
          (setq code (shell-command ecall))
          (setq ecall (concat "echo \"\n===\" >> "      readfilename)) ; CAUTION: soft-quotes essential
          (setq code (shell-command ecall))
          (message nil)                            ; hack to resize minibuffer to one line
          (message "  xem: dump-get-readme: echo call returned %d" code)
          (message "  xem: dump-get-readme: just wrote %d chars to file '%s'." chars readfilename))
      (message "  xem: dump-get-readme: user opt out."))
    chars ))

(defun xem-dump-get-htmls (dirname)
  "Provide a list of the html files in directory DIRNAME if required."

  ;; typical name : pro-235620_dc-1-014109-01.html
  ;; decode       : pro-HMS_ID-PID-CNT{.help}.html
  ;;                where HMS is the UTC timestamp, ID is the domain controller
  ;;                identifier, PID is the process id, and CNT is a counter
  ;; note         : the help files are naturally identical

  (interactive)
  (let* ((wildcard  (concat dirname "*.html")                      )
         (htmls     (file-expand-wildcards wildcard)              )  ; returns a list
         (prompt    nil                                           )
         (count     (length htmls)                                )
         (prompt    (format "Including the %d html files? " count) )
         (retval    ()                                            ))
    (if (> count 0)
        (if (y-or-n-p prompt)
            (setq retval htmls)))
    (message "  xem: dump-get-htmls: ")
    retval ))                                ; expose return

(defun xem-dump-info (readfilename           ; file name
                      role                   ; string
                      dirname                ; absolute
                      filename)              ; absolute
  "Collate and record system info using supplied arguments and direct calls."
  ;; a nil plist value simply prints as "nil"
  (let* ((bf        (current-buffer)                  )     ; lisp buffer object
         (leafname  (file-name-nondirectory filename) )
         (slist     ()                                )     ; string list
         (sbuff     nil                               )     ; string buffer
         (chars     0                                 ))    ; number of chars of information
```

```
    ;; update information
    (xem-system-info)                       ; recharge property list container (plist)
    (xem-model-info bf)                     ; recharge property list container (plist)
    ;; create 'slist'
    (push ""                                                    slist)
    (push (format "xem-title  : %s" (get 'xem-modinf 'title  )) slist)
    (push (format "model-file : %s" leafname                  ) slist)
    (push (format "svnversion : %s" (get 'xem-sysinf 'svnver )) slist)
    (push (format "dump-role  : %s" role                      ) slist)
    (push (format "used-svn   : %s" (get 'xem-modinf 'usedsvn)) slist)
    (push (format "dirname    : %s" dirname                   ) slist)
    (push (format "timestamp  : %s" (get 'xem-sysinf 'zstamp )) slist)
    (push (format "dump-uuid  : %s" (get 'xem-sysinf 'uuid   )) slist)
    (push ""                                                    slist)
    (push (format "host       : %s" (get 'xem-sysinf 'host   )) slist)
    (push (format "system     : %s" (get 'xem-sysinf 'system )) slist)
    (push (format "user       : %s" (get 'xem-sysinf 'user   )) slist)
    (push (format "email      : %s" (get 'xem-sysinf 'email  )) slist)
    (push (format "timezone   : %s" (get 'xem-sysinf 'tz     )) slist)
    (push ""                                                    slist)
    (push (format "TERM       : %s" (get 'xem-sysinf 'TERM   )) slist)
    (push (format "USER       : %s" (get 'xem-sysinf 'USER   )) slist)
    (push (format "SYNK       : %s" (get 'xem-sysinf 'SYNK   )) slist)
    (push (format "XEONA      : %s" (get 'xem-sysinf 'XEONA  )) slist)
    (push (format "XEONAR     : %s" (get 'xem-sysinf 'XEONAR )) slist)
    (push ""                                                    slist)
    ;; unpack 'slist'
    (when slist
      (dolist (s (reverse slist))
        (setq sbuff (concat sbuff "\n" s)))             ; CAUTION: 'car' not need on 's'
      (setq sbuff (substring sbuff (length "\n") nil)))     ; remove leading separator
    (message "%s" sbuff)                                    ; note quotes
    ;; write out 'slist'
    (setq chars (length sbuff))
    (setq ecall (concat "echo \"" sbuff "\" >> " readfilename)) ; CAUTION: soft-quotes are essential
    (setq code (shell-command ecall))
    (message nil)                           ; hack to resize minibuffer to one line
    ;; completion reporting
    (message "  xem: dump-info: complete.")
    (length slist) ))

(defun xem-dump ()
  "Interactive interface to function `xem-dump'.

See also the documentation for main call."
  ;; absolute names have the form "symbol*" whereas relative names do not
  (interactive)
  (let* ((filename* (buffer-file-name nil)                )  ; absolute name of current buffer
         (dirname*  (file-name-directory filename*)       )  ; base directory with trailing slash
         (filestub  (file-name-sans-extension filename*) )  ; absolute name without the ".xem"
         (fileext   (file-name-extension filename*)       )  ; should be "xem"
         (filename  (file-relative-name filename*)        )
         (dirname   (file-relative-name dirname*)         )
         (filestub  (file-relative-name filestub)         )
         (tarstub   nil                                   )
         (tarsdir   nil                                   )
         (tarname   nil                                   )
         (buffread   xem-buffer-dump-readme-text          )
         (readme    "readme.txt"                          )  ; readme file name
         (tarextz   "tgz"                                 )  ; assumes --gzip
         (boname    xem-buffer-dump-tar-capture           )
         (bo        nil                                   )
         (readname  nil                                   )  ; readme file
         (lines     0                                     )  ; information lines
         (chars     0                                     )  ; readme chars
         (stubs     ()                                    )
         (htmls     ()                                    )  ; list of html files
         (words     nil                                   )
         (tag       ""                                    )  ; string used in file and directory names
         (role      ""                                    )  ; string used in xem file stanza
         (br        nil                                   )) ; optional readme file buffer
    ;; confirm the buffer visits a XEM file
    (unless (string-equal "xem" fileext)
      (error "  xem: dump: error: call must be run from a '.%s' buffer" fileext))
    ;; seek and check tag
    (message nil)                                       ; hack to resize minibuffer
    (setq tag (xem-dump-get-tag))                       ; callout with ask
    (setq tarstub (concat filestub tag))
    (setq tarsdir (concat tarstub "/"))                 ; CAUTION: note trailing slash
    (setq tarname (concat tarstub "." tarextz))
    (if (file-exists-p tarname)
```

```
      (error "  xem: dump: error: tar file exits '%s'" tarname))
    (if (file-directory-p tarsdir)
      (error "  xem: dump: error: tar subdirectory exits '%s'" tarsdir))
    (make-directory tarsdir nil)
    ;; seek role
    (setq role (xem-dump-get-role))                    ; callout with ask
    ;; seek and save optional readme
;;; (setq readname (concat tarsdir readme))                ; simple 'readme.txt'
    (setq readname (concat tarsdir filestub "." readme))    ; qualified 'test-00.readme.txt'
    (setq lines (xem-dump-info readname role           ; callout modifies readme
                               dirname* filename*))
    (setq chars (xem-dump-get-readme readname))        ; callout with ask, modifies readme
    ;; ask about including htmls if present
    (setq htmls (xem-dump-get-htmls dirname))          ; callout with asks
    ;; save logs
    (xem-buffers-visit-all)
    ;; identify and copy files
    (setq loops 0)
    (setq stubs (file-expand-wildcards (concat filestub ".*")))  ; returns a list
    (setq files (append stubs htmls))
    (dolist (file files)
      (incf loops)
      (copy-file file tarsdir                ; copy to directory is supported
                 1                           ; '1' is confirm overwrite
                 t                           ; 't' is retain modification
                 t))                         ; 't' is preserve ownerships
    (message "  xem: dump: copied %d files to '%s'" loops tarsdir)
    ;; confirm (this stanza could be improved)
    (get-buffer-create buffread)
    (set-buffer buffread)
    (insert-file-contents readname)      ; 'nil' means do not also visit
    (switch-to-buffer buffread)
    (delete-other-windows)
    (sit-for 2)
;;; could work with a wait for close (find-file readname) (rename-buffer buffread nil) (sit-for 30)
    ;; tar archive call
    (setq tcall (concat "tar --create --gzip --verbose --file " tarname " " tarstub))
    (message "  xem: dump: tar call '%s'" tcall)
    (setq bo (get-buffer-create boname))
    (setq code (shell-command tcall bo nil)) ; 'nil' means mingle stdout and stderr
    (sit-for 2)                              ; one is too short
    (cond
     ((= code   0) (message "  xem: dump: tar returned success"))
     ((= code   1) (message "  xem: dump: tar returned some files differ error %d" code))
     ((= code   2) (message "  xem: dump: tar returned fatal error %d" code))
     (t            (message "  xem: dump: tar returned subprocess error %d" code)))
    ;; make the tarball readonly
    (set-file-modes tarname #o440)           ; CAUTION: not the unusual octal syntax
    ;; delete branch (unfortunately this version of
    ;; 'delete-directory' does not recurse, so the files need to
    ;; be deleted first
    (dolist (file files)
      (setq target (concat tarsdir (file-name-nondirectory file)))
      (delete-file target))
    (if (file-exists-p readname)
        (delete-file readname))
    (delete-directory tarstub)
    ;; report on tar file
    (setq fats (file-attributes tarname 'string))
    (message "  xem: dump: confirming tarfile '%s' with %d bytes" tarname (nth 7 fats))
    ;; completion reporting
    (message "\
  xem: dump: complete using:\n\
      tag     : '%s'\n\
      role    : '%s'\n\
      readme  : %d chars\n\
      tarfile : %s\n\
      tar     : %s"
             tag role chars tarname (xem-tar-interpret code)) ))

;  $Id: xem-2.el 5181 2010-08-24 13:07:53Z robbie $
;  end of file
```

```
;;; xem-3.el --- additional code 3 for 'xem.el'

;  file-purpose     : emacs additional code 3 for 'xem.el'
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Tue 24-Aug-2010 09:52 UTC
;  file-status      : ongoing
;  file-keywords    : emacs xeona

;  $Revision: 5181 $
;  $Date: 2010-08-24 15:07:53 +0200 (Tue, 24 Aug 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xem-3.el $

;; ---------------------------------
;;  system information
;; ---------------------------------

(defun xem-system-info ()
  "Generate a plist of standard system information."

  ;; a property list container ("plist" for short) is a list of
  ;; paired elements -- here each of the pairs associates a
  ;; property name symbol with a string value (other possibilities
  ;; exist)
  ;;
  ;; function 'getenv' returns 'nil' if ENVAR is undefined, "" if
  ;; set but null, and otherwise a meaningful string

  (let* ((xem-sysinf  ()                                                                  ) ; plist
         (tzstr       (car (cdr (current-time-zone)))                                     )
         (zformat      "%a %d-%b-%Y %H:%M UTC %z"                                         )
         (zstampstr   (format-time-string zformat (current-time) t)                      )
         (uuidstr     (replace-regexp-in-string "\n" "" (shell-command-to-string "uuidgen")) ))

    (put 'xem-sysinf  'email       user-mail-address    )   ; "robbie@actrix.co.nz"
    (put 'xem-sysinf  'host        (system-name)        )   ; "hinau.fb10.tu-berlin.de"
    (put 'xem-sysinf  'svnver      (xem-svnversion)     )   ; "0000:0000M" to "0000" or interpretation
    (put 'xem-sysinf  'system      system-configuration )   ; "x86_64-pc-linux-gnu" [1]
    (put 'xem-sysinf  'tz          tzstr                )   ; "CEST"
    (put 'xem-sysinf  'user        user-login-name      )   ; "robbie"
    (put 'xem-sysinf  'zstamp      zstampstr            )   ; [2]
    (put 'xem-sysinf  'uuid        uuidstr              )   ; 3331e6c1-9a9e-453f-abfd-163a9c3f93a6

    (put 'xem-sysinf  'SYNK        (getenv "SYNK")      )   ; "synk"
    (put 'xem-sysinf  'TERM        (getenv "TERM")      )   ; "dumb"
    (put 'xem-sysinf  'USER        (getenv "USER")      )   ; "robbie"
    (put 'xem-sysinf  'XEONA       (getenv "XEONA")     )   ; ".../xeona/svn2/futz/trunk/xeona1"
    (put 'xem-sysinf  'XEONAR      (getenv "XEONAR")    )   ; ".../xeona/svn2/futz/trunk/xeonar"

    ;; [1] note that variable 'system-type' holds a symbol
    ;; [2] format: Sun 04-Aug-2005 13:39 UTC +0000

    (message "  xem: system-info: complete.") ))

(defun xem-svnversion ()
  "Obtain current svn version and reinterpret if necessary."
  (interactive)
  (let* ((svncall    "svnversion"     )
         (xeona      (getenv "XEONA") )      ; may not be set
         (loops      0                )      ; nibble counter
         (nibble     nil              )      ; used in nibble exercise
         (redwcpath  nil              )      ; reduced subversion WC_PATH, a result of nibbling
         (value      nil              ))     ; generic value
    ;; confirm svnversion
    (setq code (shell-command (concat "which " svncall)))
    (if (not (= code 0))
        (error "  xem: subversion: error: utility not found '%s'" svncall))
    ;; nibble back 'xeona' to reveal working copy base (I could
    ;; not find a subversion call which could provide this
    ;; information)
    (setq xeona (replace-regexp-in-string "/$" "" xeona)) ; trim any trailing slash
    (setq nibble xeona)
    (when nibble
      (while (not (string-equal
                   "exported\n"             ; CAUTION: trailing newline match essential
                   (shell-command-to-string (concat svncall " " nibble))))
        (incf loops)
        (setq redwcpath nibble)
        (setq nibble (file-truename (concat nibble "/.."))) ; cute huh!
        ;; protection for bad programming
        (if (string-equal redwcpath "/")
```

```
            (error "  xem: subversion: error: root directory trip: loop count %d, redwcpath '%s'"
                   loops redwcpath))
        (if (>= loops 256)
            (error "  xem: subversion: error: loop count trip: loop count %d, redwcpath is '%s'"
                   loops redwcpath))))
    (message "\
  xem: svnversion: envar XEONA     '%s'\n\
                   reduced WC_PATH '%s'\n\
                   %d loops"
             xeona redwcpath loops)
    ;; call svnversion again
    (if redwcpath
        (setq svncall (concat svncall " " redwcpath))
      (setq svncall (concat svncall)))
    (setq value (shell-command-to-string svncall))     ; CAUTION: includes trailing newline
    (setq value (replace-regexp-in-string "\n" "" value))
    ;; reinterpret as required
    (cond ((string-equal value "")
           (setq value (concat "utility not found '" svncall "'")))
          ((string-equal value "exported")
           (setq value "envar 'XEONA' not set or call made from outside subversion working copy")))
    (message "  xem: svnversion: returning '%s' for '%s'." value redwcpath)
    value ))

;; ---------------------------------
;;  model information
;; ---------------------------------

;; general comment: the use of unqualified field names is great
;; practice -- it would be better to properly parse the model
;; file in the same way that the R scripts do

(defun xem-model-info (xembuf)
  "Interrogate the buffer XEMBUF for key data and generate a plist."
  ;; see also function `xem-system-info'
  (let* ((bfname        (buffer-name bf)  )  ; just for reporting
         (not-run-tag   "(model not run)" )
         (xem-modinf    ()                )  ; plist
         (temp          nil               ))
    ;; integrity check
    (if (string-match "\\.xem$" bfname)
        (message "  xem: model-info: commencing with data recovery from model '%s'" bfname)
      (error "  xem: model-info: error: given buffer inconsistent with model file '%s'" bfname))
    ;; continue
    (set-buffer bf)                           ; defensive programming, probably not needed
    ;; recover data and load plist
    (put 'xem-modinf 'title   (xem-value-get "r-title"  t)) ; 'program.r-processing.r-title'
    (setq temp (xem-value-get "used-svn" nil))              ; 'program.program-last-run.used-svn'
    (if (= (length temp) 0) (setq temp not-run-tag))        ; overwrite
    (put 'xem-modinf 'usedsvn temp)
    ;; completion reporting
    (message "  xem: model-info: complete.") ))

;; ---------------------------------
;;  review dumps
;; ---------------------------------

;; tar --extract --gzip --to-stdout --file="stub.tgz" "stub/readme.txt" | head -10

(defconst xem-buffer-dump-review
  "*xem-dump-review*"
  "Buffer for dump reviews.")

(defun xem-dump-summarize ()
  "Scan and summarize dump files."
  (interactive)
  (let* ((buffname   xem-buffer-dump-review          )
         (readhead   10                              )
         (readname   "readme.txt"                    )
         (tars       (file-expand-wildcards "*.tgz") )
         (loops      0                               )
         (leaf       nil                             )
         (timestamp  nil                             )
         (timelist   nil                             )
         (age        nil                             )
         (readme     nil                             )
         (plural     ""                              ))
    ;; create buffer
    (switch-to-buffer (get-buffer-create buffname))
    (delete-other-windows)
    ;; add line if needed
```

```
    (unless (= (point) (point-min))
      (insert "\n===\n\n"))
    ;; load
    (dolist (tar tars)
      (setq target (concat (file-name-sans-extension tar) "/" readname))
      (setq tcall (concat "tar --extract --gzip --to-stdout --file=" tar " " target))
      (setq tcall (concat tcall " | " "head --lines " (number-to-string readhead)))
      (setq readme (shell-command-to-string tcall))
      ;; check for tar output
      (if (string-match "^tar:" readme)
          (progn
            (insert "\n")
            (insert tcall)
            (insert readme)
            (insert "\n")
            (message "xem: dumps-summarize: tar call '%s' failed " tcall))
        (incf loops)
        (insert "\n")
        (setq leaf (file-name-sans-extension tar))
        (insert (format "%03d  %s\n" loops leaf))
        (insert "\n")
        (insert (xem-dump-extract "xem-title"   readme 'line t))
        (insert (xem-dump-extract "model-file"  readme 'line t))
        (insert (xem-dump-extract "svnversion"  readme 'line t))
        (insert (xem-dump-extract "used-svn"    readme 'line t))
        (insert (xem-dump-extract "dump-role"   readme 'line t))
        (insert (xem-dump-extract "timestamp"   readme 'line t))
        ;; age code
        (setq timestamp (xem-dump-extract "timestamp" readme 'value))
        (setq timelist (xem-time-decomp timestamp))
        (setq age (xem-time-delta timelist))
        (insert (format "dump-age   : %s\n" age))
        (insert "\n")))
    ;; summarize
    (insert "\n---\n\n")
    (if (= loops 1) (setq plural "") (setq plural "s"))
    (insert (format "%d dump file%s processed\n" loops plural))
    (insert "\n")
    (insert
     "tarbomb : tar -extract --verbose --verbose --gzip --strip-components=1 --overwrite --file\n")
    (insert
     "tarball : tar -extract --verbose --verbose --gzip --strip-components=0 --overwrite --file\n")
    (message "  xem: dumps summarize complete with %d readmes from %d dumps." loops (length tars))
    loops ))

;; good time code in: http://www.emacswiki.org/emacs/download/rmail-extras.el

;; ---------------------------------
;;  time functions
;; ---------------------------------

(defun xem-time-decomp (timestamp)
  "Turn the formatted TIMESTAMP into a standard time value.

The TIMESTAMP is assumed to be in the form \"%a %d-%b-%Y %H:%M UTC +0000\".
This code uses `encode-time' and the resulting list ranges from seconds to year.
Here are some representative intermediate and final results:

   timestamp 'Sat 21-Aug-2010 15:04 UTC +0000'
   decomp '(Sat 21 Aug 2010 15 04 UTC +0000)'
   split '00 04 15 21 08 2010'
   encode '(19567 60128)'"
  (let* ((sep        "[ :-]"                       )  ; CAUTION: "-" at end, else range operator
         (decomp     (split-string timestamp sep t) )
         (day        (nth 1 decomp)                 )
         (monstr     (nth 2 decomp)                 )
         (year       (nth 3 decomp)                 )
         (hour       (nth 4 decomp)                 )
         (min        (nth 5 decomp)                 )
         (sec        "00"                           )
         (zonestr    (nth 6 decomp)                 )
         (offsetstr  (nth 7 decomp)                 )  ; no used
         (monthmap   '(("Jan" . "01") ("Feb" . "02") ("Mar" . "03") ("Apr" . "04")
                       ("May" . "05") ("Jun" . "06") ("Jul" . "07") ("Aug" . "08")
                       ("Sep" . "09") ("Oct" . "10") ("Nov" . "11") ("Dec" . "12")))
         (mon        nil                            )
         (ret        nil                            ))
    (if (not (string-equal zonestr "UTC"))
        (error "  xem: time-decomp: error: timestamp not UTC: '%s'" zonestr))
    (setq mon (cdr (assoc monstr monthmap)))
    (message "  xem: time-decomp: timestamp '%s'" timestamp)
```

```
    (message "  xem: time-decomp: decomp '%s'" decomp)
    (message "  xem: time-decomp: split '%s %s %s %s %s %s'" sec min hour day mon year)
    (setq day  (string-to-number day ))
    (setq mon  (string-to-number mon ))
    (setq year (string-to-number year))
    (setq hour (string-to-number hour))
    (setq min  (string-to-number min ))
    (setq sec  (string-to-number sec ))
    (setq ret (encode-time sec min hour day mon year t))    ; CAUTION: t is essential for UTC
    (message "  xem: time-decomp: encode '%s'" ret)
    (message "  xem: time decomp complete.")
    ret ))                                  ; expose return value

(defun xem-time-delta (timelist)
  "Interpret the difference between TIMELIST and the current time.

It is assumed that TIMELIST is expressed as UTC and not in the local timezone.
An interpretation of \"3 days\" means 3 to 4 days."
  ;; CAUTION: this code developed and tested on a 64-bit machine,
  ;; more specifically the LP64 data model
  (let* ((now        nil )
         (was        nil )
         (epoch-now  nil )
         (epoch-was  nil )
         (epoch-dif  nil )
         (delta      nil )
         (msg        nil ))
    ;; establish time points
    (setq now (current-time))
    (setq was timelist)
    ;; convert to floating point seconds since UNIX epoch (starting 1970)
    (setq epoch-now (float-time now))
    (setq epoch-was (float-time was))
    (setq epoch-dif (- epoch-now epoch-was))
    ;; report
    (message "  xem: time-delta: epoch-now %s" epoch-now)
    (message "  xem: time-delta: epoch-was %s" epoch-was)
    (message "  xem: time-delta: epoch-dif %s" epoch-dif)
    ;; interpret
    (when t
      (setq delta (/ epoch-dif (* 60 60 24)))
      (setq delta (truncate delta))            ; convert to integer by rounding down
      (cond
       ((= delta 0) (setq msg "under one day"))
       ((= delta 1) (setq msg "one day"))
       (t           (setq msg (format "%d days" delta)))))
    (when (= delta 0)
      (setq delta (/ epoch-dif (* 60 60)))
      (setq delta (truncate delta))
      (cond
       ((= delta 0) (setq msg "under one hour"))
       ((= delta 1) (setq msg "one hour"))
       (t           (setq msg (format "%d hours" delta)))))
    (when (= delta 0)
      (setq delta (/ epoch-dif (* 60)))
      (setq delta (truncate delta))
      (cond
       ((= delta 0) (setq msg "under one minute"))
       ((= delta 1) (setq msg "one minute"))
       (t           (setq msg (format "%d minutes" delta)))))
    ;; completion reporting
    (message "  xem: time delta complete with interpretation = %s." msg)
    msg ))                                  ; expose return value

;; ---------------------------------
;;  utilities
;; ---------------------------------

(defun xem-dump-extract (key readtext type &optional newline)
  "Hunt for KEY in READTEXT, return TYPE, possibly with trailing NEWLINE.

Note TYPE in { 'line 'value } and NEWLINE in { t nil }."
  (let* (
         (val-trip  (concat "^" key "[[:blank:]]*:[[:blank:]]\\(.*\\)$") )
         (lin-trip  (concat "^\\(" key ".*\\)$")                         )
         (symname   (symbol-name type)                                   ))
    ;; select a 'trip' regex
    (cond
     ((equal type 'value) (setq trip val-trip))
     ((equal type 'line)  (setq trip lin-trip)))
    ;; attempt match
```

```
    (if (string-match trip readtext)
        (progn
          (setq ret (match-string 1 readtext))
          (message "  xem: dumps-extract: key '%s' yielded match '%s'" key ret)
          ;; add newline if requested
          (if newline
              (progn
                (setq ret (concat ret "\n"))
                (message "  xem: dumps-extract: newline added")))
          (message "  xem: dumps extract successful for key '%s' and trip regex '%s'." key trip))
      (setq ret "")
      (message "  xem: dumps extract failed for key '%s' and trip regex '%s'" key trip))
    ret ))                                  ; expose return value

;; ---------------------------------
;;  tarball functions
;; ---------------------------------

(defun xem-dump-unpack (dumpfile*)          ; the trailing * indicates a (probable) absolute name
  "Unpack dump file DUMPFILE* (the star is part of the symbol identifier).

This function first asks for a '.tgz' file, which is tested.  And then
whether to \"extract into subdir or bomb\".  Extract into subdir is the
safer option.  If bomb is chosen and this function unpacks a model file
which is also open, then that buffer will be reverted."
  (interactive "fSelect a '.tgz' file (tested): ")
  (let* ((ext       (file-name-extension dumpfile*)            )
         (dumpfile  (file-name-nondirectory dumpfile*)         )
         (ref       "tgz"                                      )
         (prompt    "Extract into subdir (yes) or bomb (no)? " )
         (tcall     nil                                        )
         (level     nil                                        )
         (code      nil                                        )
         (msg       nil                                        ))
    (unless (string-equal ext ref)
      (error "  xem: dump-unpack: extension '.%s' must be '.%s', abandoning without action" ext ref))
    (if (y-or-n-p prompt)
        (setq level 0)
      (setq level 1))
    (setq tcall (format "tar --extract --verbose --verbose --gzip \
--strip-components=%d --overwrite --file %s" level dumpfile))
    (setq code (shell-command tcall nil t))
    ;; competion reporting
    (message "\
  xem: dump unpack tar call '%s'\n\
       tar  %s" tcall (xem-tar-interpret code)) ))

;; ---------------------------------
;;  clean functions
;; ---------------------------------

(defun xem-dump-clean-buffer ()
  "Prompted en-masse clean up of files (not subdirs) related to current buffer.

This function is a wrapper to `xem-dump-clean'."
  (interactive)
  (let* ((filename*  (buffer-file-name nil)))               ; absolute name of current buffer
         (xem-dump-clean filename*) ))                      ; lack of 'message' intentional

(defun xem-dump-clean (refmodel)
  "Prompted en-masse clean up of files (but not subdirectories) related to REFMODEL."
  (let* ((filename* refmodel                              )  ; should be absolute (not tested)
         (dirname*  (file-name-directory filename*)        )  ; base directory with trailing slash
         (filestub  (file-name-sans-extension filename*) )  ; absolute name without the ".xem"
         (fileext   (file-name-extension filename*)       )  ; should be "xem"
         (filename  (file-relative-name filename*)        )
         (dirname   (file-relative-name dirname*)         )
         (filestub  (file-relative-name filestub)         )
         (skips     "\\.xem\\'\\|.tgz\\'"                 )  ; file patterns to keep
         (pattern   nil                                   )  ; file match wildcard
         (matches   ()                                    )  ; list of files which match
         (deletes   ()                                    )  ; list of files to be deleted
         (count     nil                                   )  ; number of deletes
         (prompt    nil                                   )  ; dynamic prompt for `y-or-n-p'
         (loops     0                                     )) ; delete file counter
    ;; confirm the buffer visits a XEM file
    (unless (string-equal "xem" fileext)
      (error "  xem: dump-clean: error: call must be run from a '.%s' buffer" fileext))
    (setq pattern (concat filestub "*"))
    (setq matches (file-expand-wildcards pattern nil))      ; 'nil' for relative paths
    ;; exclude some
```

```
    (dolist (file matches)
      (unless (string-match skips file)                    ; note the "\'" end-of-buffer operator
        (push file deletes)))
    (message "  xem: dump-clean: pattern '%s' yields %d matches and %d deletes"
             pattern (length matches) (length deletes))
    (setq count (length deletes))
    (cond
     ((< count 0)
      (error "  xem: dump-clean: error: coding issue, encountered negative count %d" count))
     ((= count 0)
      (message "  xem: dump clean found %d matches, exiting without action." count))
     ((> count 0)
      (setq prompt (format "Pattern '%s' matched %d files, continue? " pattern count))
      (if (y-or-n-p prompt)
          (progn
            (dolist (file deletes)
              (incf loops)
              (delete-file file))
            (message "  xem: dump clean compete with %s deleted files." loops))
       (message "  xem: dump clean abandoned task without action." loops)))) ))

;  $Id: xem-3.el 5181 2010-08-24 13:07:53Z robbie $
;  end of file
```

```
;;; xrstat.el --- create and call R stub programs for visualization

;  file-purpose     : emacs create and call R stub programs for visualization
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Thu 12-Aug-2010 22:07 UTC
;  file-status      : ongoing
;  file-keywords    : emacs xeona

;  $Revision: 5202 $
;  $Date: 2010-09-21 16:02:30 +0200 (Tue, 21 Sep 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xrstat.el $

;  XEONA CODEBASE COPY
;
;  * this code originated as part of the xeona codebase
;
;  * the GLPv2 license included here derives from Emacs Lisp
;    coding practice and from not the 'xeona' project --
;    users can optionally chose to adopt the 'xeona' norm
;    instead

;; Copyright (C) 2010 Robbie Morrison

;; This file is *NOT* part of GNU Emacs.

;; This file is free software; you can redistribute it
;; and/or modify it under the terms of the GNU General
;; Public License as published by the Free Software
;; Foundation; either version 2, or (at your option)
;; any later version.

;; This file is distributed in the hope that it will be
;; useful, but WITHOUT ANY WARRANTY; without even the
;; implied warranty of MERCHANTABILITY or FITNESS FOR A
;; PARTICULAR PURPOSE.  See the GNU General Public
;; License for more details.

;; You may have received a copy of the GNU General
;; Public License along with GNU Emacs; see the file
;; COPYING.  If not, write to the Free Software
;; Foundation, Inc., 59 Temple Place - Suite 330,
;; Boston, MA 02111-1307, USA.

;;; Commentary:

;;  TODOs
;;
;;      > hold open displays (they flash for an instant)
;;      > xrstat-mode with line highlights for "^>" and "^xeona"
;;
;;  Known issues
;;
;;  Background
;;
;;      reference: An Introduction to R / B.4 Scripting with R
;;      reference: An Introduction to R / B.1 Invoking R from the command line
;;      web: http://cran.r-project.org/doc/manuals/R-intro.html
;;
;;  Approaches
;;
;;      there appear to be several approaches, whereby 'file.R'
;;      contains <expressions>
;;
;;          $ R -e <expression> -e <expression>
;;          $ echo <expressions> | R --file -
;;          $ R --file temp.R
;;          $ R CMD BATCH temp.R
;;          - 'Rscript'
;;
;;  Typical consolidated command
;;
;;      R --no-save --no-restore --quiet --verbose
;;        -e 'source("~/synk/xeona/svn2/futz/trunk/xeonar/xem.R")'
;;        -e 'xem.job("~/synk/xeona/svn2/futz/trunk/models/test-16.xem")'

;;; History:

;;      * see the subversion log messages

;;; Code:
```

```
;; ================================
;;  xem support
;; ================================

(defconst xem-r-dir (if (getenv "XEONAR")
                        (getenv "XEONAR")
                      (concat "/home/robbie/synk/xeona/svn2/futz/trunk/xeonar/")))

(defconst xem-buffer-r-capture "*r-capture*" "Buffer for R output capture.")

;; --------------------------------
;;  R program call
;; --------------------------------

(defun xem-r-run ()
  "Run the 'xem.R' program on the current buffer."
  (interactive)
  (let* ((bm        (current-buffer)                          )   ; xem file buffer
         (bc        (get-buffer-create xem-buffer-r-capture) )   ; output capture buffer
         (target    (buffer-file-name nil)                    )   ; file associated with current buffer
         (binary    xem-binary                                )   ; xeona binary
         (xeona     nil                                       )   ; xeona binary with quotes or NULL
         (rbin      "R"                                       )
         (rdir      xem-r-dir                                 )
         (rscript   "xem.R"                                   )
         (dbug      1                                         )   ; debug reporting in { 0 1 2 3 4 }
         (rfile     nil                                       )   ; rdir + rscript
         (jargs     ()                                        )
         (jarg      nil                                       )   ; xem.job arguments (concatenated)
         (rcmds     ()                                        )
         (rcmd      nil                                       )   ; R program (concatenated)
         (ropts     ()                                        )
         (ropt      nil                                       )   ; R options (concatenated)
         (rcall     nil                                       )
         (code      nil                                       ))
    ;; outset reporting (because this is a long set of procedures)
    (message "  xem: r-run: commencing with target '%s'" target)
    ;; check if run -- otherwise the R code errors out as follows:
    ;;       info : xem.job : plot list job starts
    ;;     Error in xy.coords(x, y, xlabel, ylabel, log) :
    ;;       'x' and 'y' lengths differ
    ;;     Calls: xem.job -> xem.plotValue -> plot -> plot.default -> xy.coords
    (if (xem-have-i-run)                    ; active call
        (message "  xem: r-run: buffer identified as having run")
      (error "  xem: r-run: error: buffer identified as NOT having run"))
    ;; identify model or guard and set 'xeona' appropriately
    (setq binary (abbreviate-file-name binary))
    (if (string-match "\\.guard\\.xem$" target)
        (setq xeona (concat "\"" binary "\""))
      (setq xeona "NULL"))                                    ; CAUTION: no quoting
    ;; assemble R script name
    (if (not (string-equal (substring rdir -1 nil) "/"))     ; cannot guarantee a trailing slash
        (setq rdir (concat rdir "/")))
    (setq rfile (concat rdir rscript))
    (setq rfile (abbreviate-file-name rfile))
    ;; create 'job' call
    (setq target (abbreviate-file-name target))
    (push (concat "xemfile=\"" target "\""              ) jargs)
    (push (concat "xeona=" xeona                        ) jargs)
    (push (concat "dbug=\"" (number-to-string dbug) "\"") jargs)
    (setq jarg (xem-stringify-list jargs ", "))
    ;; create "program"
    (push (concat "-e " "'source(\"" rfile "\")'" ) rcmds)  ; load script command
    (push (concat "-e " "'xem.job(" jarg ")'"     ) rcmds)  ; run function command
    (setq rcmd (xem-stringify-list rcmds))
    (message "  xem: r run: rcmd '%s'" rcmd)
    ;; R and R options
    (push "--no-save"     ropts) ; do NOT save workspace at the end of the session
    (push "--no-restore"  ropts) ; do NOT restore previously saved objects, the R history file, et al
;;; (push "--vanilla"     ropts) ; --no-save, --no-restore, --no-site-file, --no-init-file --no-environ
    (push "--quiet"       ropts) ; omit startup message
;;; (push "--slave"       ropts) ; make R run as quietly as possible
;;; (push "--verbose"     ropts) ; print more information about progress
;;; (push "--interactive" ropts) ; CAUTION: not required
;;; (setq ropt (combine-and-quote-strings (reverse ropts)))
    (setq ropt (xem-stringify-list ropts))
    (message "  xem: r run: ropt '%s'" ropt)
    ;; assemble call
    (setq rcall (concat rbin " " ropt " " rcmd))
    (message "  xem: r run: rcall '%s'" rcall)
```

```
    ;; confirm R
    (setq code (shell-command "R --version"))
    (sit-for 1)
    (delete-other-windows)
    (cond
     ((= code   0) (message "  xem: r-run: R command found"))
     ((= code 127) (error "  xem: r-run: R command not found, expected exit code %d" code))
     (t            (error "  xem: r-run: R command unexpected exit code %d" code)))
    ;; confirm R script
    (access-file rfile "  xem: r-call: abandoning task without action")
    (message "  xem: r run: access file passed '%s'" rfile)
    ;; prompted save if required ('bm' assumed here)
    (if (buffer-modified-p)
        (if (y-or-n-p "Save this model in order to continue? ")
            (save-buffer)
          (error "  xem: r run call abandoning '%s' R call without action" rcall)))
    (save-buffer)                                       ; defensive programming, not really needed
    ;; prepare output buffer
    (set-buffer bc)                                     ; 'bc' is the capture buffer
    (xrstat-mode t)                                     ; minor mode, 't' is activate
    (erase-buffer)                                      ; delete entire contents, ignores any narrowing
    (message "  xem: r run: buffer '%s' erased" (buffer-name bc))
;;; (error "  xem: r-call: deliberate exit")
    ;; run command
    (setq code (shell-command rcall bc nil))            ; 'nil' means mingle stdout and stderr
    (message "  xem: r-call: R ran synchronously and returned %s" code)
    (xrstat-summarize-buffer bc)                        ; create a summary
    (message "  xem: r call = '%s'\n       r exit = %d" rcall code) ))

;; ---------------------------------
;;  utilities
;; ---------------------------------

(defun xem-stringify-list (slist &optional sep)
  "Stringify string SLIST using optional SEP if provided, else \" \"."
  (let* ((buffer  "" ))                             ; returns an empty string if 'slist' is nil
    (if (not sep) (setq sep " "))
    (message "  xem: stringify-list: sep '%s'" sep)
    (message "  xem: stringify-list: length %d" (length sep))
    (message "  xem: stringify-list: slist '%s'" slist)
    (message "  xem: stringify-list: length %d" (length slist))
    (when slist
      (dolist (s (reverse slist))
        (setq buffer (concat buffer sep s)))  ; CAUTION: 'car' not need on 's'
      (setq buffer (substring buffer (length sep) nil)))
    buffer ))                                       ; expose return value

;; ---------------------------------
;;  open plot window files
;; ---------------------------------

;; typical viewers (bad results)
;;
;;   png : display emacs eog gthumb
;;   pdf : acroread emacs evince
;;   svg : (display) eog evince firefox gthumb inkscape
;;
;;   notes - eog gets confused with multiple files

(defun xem-r-pngs ()
  "Display any PNG files associated with the current model."
  (interactive)
  (let* ((bc    (current-buffer)       )
         (call  "gthumb --fullscreen"  )     ; works well for dedicated viewing
         (extn  "png"                  ))
    (message "  xem: r-pngs: extn = %s  buffer = %s  call = %s" extn (buffer-name bc) call)
    (xrstat-files bc call extn)
    (message "  xem: r-pngs: complete using extension '%s' and call '%s'" extn call) ))

(defun xem-r-pdfs ()
  "Display any PDF files associated with the current model."
  (interactive)
  (let* ((bc    (current-buffer)       )
         (call  "evince --fullscreen"  )
         (extn  "pdf"                  ))
    (xrstat-files bc call extn)
    (message "  xem: r-pdfs: complete using extension '%s' and call '%s'" extn call) ))

(defun xem-r-svgs ()
  "Display any SVG files associated with the current model."
  (interactive)
```

```
  (let* ((bc    (current-buffer)       )
         (call  "gthumb --fullscreen"  )
         (extn  "svg"                  ))
    (xrstat-files bc call extn)
    (message "  xem: r-svgs: complete using extension '%s' and call '%s'" extn call) ))

(defun xrstat-files (buffer                   ; lisp buffer object
                     util                     ; viewer utility call string
                     ext)                     ; file extension without dot
  "View files associated with BUFFER using call UTIL and filtering on extension EXT."
  ;; CAUTION: this code breaks if matched filenames contain
  ;; whitespace (which should not be the case)
  (let* ((model    (buffer-file-name buffer)                     )
         (stub     (xrstat-stub-me model)                        )   ; grab stub name
         (pattern  (concat stub ".*." ext)                       )
         (files    (file-expand-wildcards pattern t)             )   ; used to confirm some files
         (call     (concat util " " "$(ls -tr " pattern ") " "&") )   ; NOTE: "&" for asynchronous call
         (code     nil                                           ))
    (if files
        (progn
          (message "  xrstat: files: hits = %d" (length files))
          (message "  xrstat: files: matching files:")
          (dolist (file files)
            (message "    %s" (file-name-nondirectory file)))
          (message "  xrstat: files: call = '%s'" call)
          (shell-command call nil nil)      ; CAUTION: no return status for asynchronous call
          (message "  xrstat: files: complete with base call '%s'." call))
      (message "  xrstat: files: no files match wildcard pattern '%s'." pattern)) ))

(defun xrstat-stub-me (filename)                ; normal leaf, relative, or absolute
  "Stub the FILENAME."
  (let* ((filename  (abbreviate-file-name     filename) )
         (buf-1     (file-name-sans-extension filename) )
         (ext-1     (file-name-extension      filename) )   ; should be "xem"
         (buf-2     (file-name-sans-extension buf-1)    )
         (ext-2     (file-name-extension      buf-1)    )   ; could possibly be "guard"
         (ret       nil                                 ))
    (cond
     ((string-equal "guard" ext-2) (setq ret buf-2))
     ((string-equal "xem"   ext-1) (setq ret buf-1))
     (t                            (setq ret filename)))
    (message "  xrstat: stub-me: processing '%s' to '%s'" filename ret)
    ret ))                                      ; stubbed filename

;; ---------------------------------
;;  provide
;; ---------------------------------

(provide 'xrstat)

;; =================================
;;  xrstat-mode minor mode
;; =================================

;; ---------------------------------
;;  requirements
;; ---------------------------------

(require 'hi-lock)                              ; "minor mode for interactive automatic highlighting"

;; ---------------------------------
;;  key map
;; ---------------------------------

;; CAUTION: `xrstat-mode-map' should precede `define-minor-mode'

(defvar xrstat-mode-map nil "Local keymap for Rifs mode buffers.")

(if xrstat-mode-map
    ()                                          ; protection against double loading
  ;; key bindings
  (setq xrstat-mode-map (make-sparse-keymap))
  (define-key xrstat-mode-map [(shift        prior)] 'xrstat-move-regex-prior)
  (define-key xrstat-mode-map [(shift         next)] 'xrstat-move-regex-next))

;; ---------------------------------
;;  minor mode
;; ---------------------------------

;;;###autoload(defvar xrstat-mode nil)
(define-minor-mode xrstat-mode
```

```
  "Minor mode for displaying R output capture.

This mode highlights R calls and highlights warnings and errors,
enables navigation, and offers a summarize function.
Key-bindings include:

  next     : \\[xrstat-move-regex-next]
  previous : \\[xrstat-move-regex-prior]

With no argument, this command toggles the mode.  A non-null
prefix argument turns on the mode.  A null prefix argument turns
off the mode."

  ;; common items

  :init-value nil
  :lighter " Xrstat"                        ; modeline lighter
  :keymap nil
  :global nil                               ; nil means buffer-local minor mode
  :group 'xrstat
  :version nil                              ; emacs version (and not mode version)

  ;; code

  (message "  xrstat: xrstat-mode: commencing")
  (if xrstat-mode
      (progn
        (visual-line-mode t)                ; employ visual not logical lines, activate 'word-wrap'
        (toggle-truncate-lines 1)
        (xrstat-colorize)
        (message "  xrstat: toggle on mode"))
    (progn
      (visual-line-mode nil)
      (toggle-truncate-lines -1)
      (hi-lock-mode nil)
      (message "  xrstat: toggle off mode")))

  ) ; final parenthesis

;; ---------------------------------
;;  navigation
;; ---------------------------------

(defconst xrstat-move-regex
  "Warning message\\|warn\\|error\\|Error\\|ERROR"
  "Xrstat move regex.")

(defconst xrstat-recenter
  nil                                       ; useful values: 0, 1, nil (to center)
  "Xrstat navigation recenter behavior.")

(defun xrstat-move-regex-next ()
  "Navigate to next regex."
  (interactive)
  (let ((regex  xrstat-move-regex ))
    (end-of-line)                           ; need to "kick-start" the search

    (if (re-search-forward regex
                           nil t)           ; limit (no), no error (true)
        (progn
          (beginning-of-line)
          (recenter xrstat-recenter)
          (beginning-of-line)
          (message "  xrstat: move to next regex complete."))
      (goto-char (point-max))
      (message "  xrstat: move to next regex FAILED."))
    (match-beginning 0) ))                  ; nil if no match

(defun xrstat-move-regex-prior ()
  "Navigate to previous regex."
  (interactive)
  (let ((regex  xrstat-move-regex ))
    (if (re-search-backward regex
                            nil t)          ; limit (no), no error (true)
        (progn
          (beginning-of-line)
          (recenter xrstat-recenter)
          (beginning-of-line)
          (message "  xrstat: move to previous regex complete."))
      (goto-char (point-min))
      (message "  xrstat: move to previous regex FAILED."))
```

```
    (match-beginning 0) ))                  ; nil if no match

;; ---------------------------------
;;  highlight and summarize
;; ---------------------------------

(defface xrstat-color-B01
  '((((background dark)) (:background "sienna1" :foreground "black"))
    (t (:background "sienna1")))
  "Private face.")

(defface xrstat-color-B02
  '((((background dark)) (:background "DarkGoldenrod1" :foreground "black"))
    (t (:background "DarkGoldenrod1")))
  "Private face.")

(defface xrstat-color-B03
  '((((background dark)) (:background "DarkGoldenrod1" :foreground "black"))
    (t (:background "DarkGoldenrod1")))
  "Private face.")

(defface xrstat-color-B04
  '((((background dark)) (:background "red" :foreground "black"))
    (t (:background "red")))
  "Private face.")

(defun xrstat-colorize ()
  "Highlight key lines and phrases."
  (interactive)
  (let* ((regex-1  "^> ."                  )
         (regex-2  "Warning message"       )
         (regex-3  "warn"                  )
         (regex-4  "error\\|Error\\|ERROR" ))
    (highlight-lines-matching-regexp regex-1 'xrstat-color-B01)
    (highlight-phrase                regex-2 'xrstat-color-B02)
    (highlight-phrase                regex-3 'xrstat-color-B03)
    (highlight-lines-matching-regexp regex-4 'xrstat-color-B04)
    (message "  xrstat: colorize: complete") ))

(defun xrstat-summarize-buffer (buffer)
  "Summarize r capture in given BUFFER."
  (interactive)
  (let* ((bname (buffer-name buffer)        )
         (regex-2  "Warning message"        )
         (regex-3  "warn"                   )
         (regex-4  "error\\|Error\\|ERROR"  )
         (regex-5  "invalid graphics state" ))
    (goto-char (point-min))
    (insert "              capture status\n")
    (insert (xrstat-format-occurs "R  w a r n"            regex-2))
    (insert (xrstat-format-occurs "xeona  w a r n"        regex-3))
    (insert (xrstat-format-occurs "all  e r r o r"        regex-4))
    (insert (xrstat-format-occurs "invalid graphic state" regex-5))
    (insert "\n")
    (message "  xrstat: summarize-buffer: complete with '%s'." bname) ))

(defun xrstat-format-occurs (tag regex)
  "Report TAG described occurrences of REGEX in current buffer."
  (save-excursion
    (let* ((line (format "  %22s : %d\n" tag (xrstat-count-occurs regex)) ))
      line )))

(defun xrstat-count-occurs (regex)
  "Count occurrences of REGEX in current buffer."
  (let* ((prior             case-fold-search )
         (case-fold-search  nil              )    ; non-nil means ignore case
         (count             0                ))
    (goto-char (point-min))
    (while (re-search-forward regex nil t)
      (incf count))
    (setq case-fold-search prior)
    (message "  xrstat: count-occurs: regex '%s' occurred %d times" regex count)
    count ))

;; ---------------------------------
;;  provide
;; ---------------------------------

;; put the mode symbol into the list "features", so that users can
;; invoke (require 'xrstat-mode) and load your code only when needed
```

```
(provide 'xrstat-mode)

;;; xrstat.el ends here

;  $Id: xrstat.el 5202 2010-09-21 14:02:30Z robbie $
;  end of file
```

```
;;; xog.el --- display support for captured xeona terminal output

;  file-purpose     : emacs display support for captured xeona terminal output
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Thu 12-Aug-2010 22:07 UTC
;  file-status      : working
;  file-keywords    : emacs xeona

;  $Revision: 5280 $
;  $Date: 2010-10-08 11:28:22 +0200 (Fri, 08 Oct 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xog.el $

;  XEONA CODEBASE COPY
;
;  * this code originated as part of the xeona codebase
;
;  * the GLPv2 license included here derives from Emacs Lisp
;    coding practice and from not the 'xeona' project --
;    users can optionally chose to adopt the 'xeona' norm
;    instead

;; Copyright (C) 2010 Robbie Morrison

;; This file is *NOT* part of GNU Emacs.

;; This file is free software; you can redistribute it
;; and/or modify it under the terms of the GNU General
;; Public License as published by the Free Software
;; Foundation; either version 2, or (at your option)
;; any later version.

;; This file is distributed in the hope that it will be
;; useful, but WITHOUT ANY WARRANTY; without even the
;; implied warranty of MERCHANTABILITY or FITNESS FOR A
;; PARTICULAR PURPOSE.  See the GNU General Public
;; License for more details.

;; You may have received a copy of the GNU General
;; Public License along with GNU Emacs; see the file
;; COPYING.  If not, write to the Free Software
;; Foundation, Inc., 59 Temple Place - Suite 330,
;; Boston, MA 02111-1307, USA.

;;; Commentary:

;;  ToDos
;;
;;      * check current buffer logic (although no problems have
;;        been detected), while noting the warning in section 27.2
;;        of the elisp manual
;;
;;      * add `abbreviate-file-name' to messages where appropriate
;;
;;      * run `checkdoc' periodically
;;
;;  Known issues
;;

;;; History:

;;      * see the subversion log messages

;;; Code:

;; ---------------------------------
;;  user-modifiable global constants
;; ---------------------------------

(defconst xog-workingcopy
  (file-truename (substitute-in-file-name "$HOME/$SYNK/xeona/svn2/futz/"))
  "Subversion working copy for 'xeona'.")

(defconst xog-sourcecode
  (file-truename (concat xog-workingcopy "trunk/xeona1/"))
  "Source code root for 'xeona'.")

(defconst xog-extension
  ".xog"
  "Xeona terminal log file extension (NOT set in 'common.cc').")
```

```
(defconst xog-recenter nil                    ; useful values: 0, 1, nil (to center)
  "Define recenter after search behavior.")

(defconst xog-file-stem
  xeona-file-stem
  "Current directory.")

(defconst xog-xumber-file
  (concat xog-file-stem "xumber.el")
  "Xumber mode file to drag in.")

;; ---------------------------------
;;  requirements
;; ---------------------------------

(require 'hi-lock)                      ; "interactive automatic highlighting"
(require 'highlight-current-line)       ; "highlight line where the cursor is"

;; note the various packages which support current line highlighting

;;; (load-library "/home/robbie/synk/xeona/svn2/futz/trunk/elisp/highline.el")
;;; (require 'highline)       ; "highlight current line in buffer" (non-std)
;;; (require 'hl-line)        ; "highlight the current line"

;; ---------------------------------
;;  mode function
;; ---------------------------------

(defun xog-mode ()
  "Major mode for displaying captured xeona terminal output.

All interactive functions start \"xog-\" for easy identification.

If the menu bar is hidden, a background Xog menu can be brought
up with: \"Ctrl-Shift-leftmouse\".

Dedicated key binding are indicated in the Xog menu.  For a
complete listing, use \\[describe-bindings]."
  (interactive)

  (message "  xog: xog-mode setup commencing")

  ;; recommended
  (kill-all-local-variables)

  ;; emacs stuff
  (setq major-mode 'xog-mode)           ; used by C-h m aka 'describe-mode'
  (setq mode-name  "Xog")
  (use-local-map    xog-mode-map)
  (run-hooks       'xog-mode-hook)      ; optionally defined in user's configs

  ;; use our custom menu, "Xog", located to the right of "Tools"
  (menu-bar-mode 1)

  ;; visual bell (because the Ubuntu system bell is broken as of mid-2010)
  (setq visible-bell t)

  ;; background (can be redefined within 'xog-mode-hook')
  (setq fill-column 999)                ; auto-fill wrap, essentially disabled
  (setq selective-display-ellipses nil) ; omit "..."
  (toggle-truncate-lines 1)             ; "1" is never fold (wrap) long lines

  ;; global variables
  ;; <none>

  ;; make your comment command use the same shortcut
  ;;   for 'comment-dwim' (dwim = do what I mean),
  ;; the user may have changed their default
  ;; (define-key xog-mode-map [remap comment-dwim] 'xog-comment-dwim)

  ;; miscellaneous
  (setq message-log-max t)              ; 't' is do not truncate "*Messages*"

  ;; general actions
  (highlight-current-line-minor-mode 1) ; highlight current line
  (xog-colorize)                        ; automatically colorize on opening
  (goto-line 3)

  ;; reset some variables
  ;;
  ;; tests indicate that these values do persist between killed
```

```
  ;; (‘kill-buffer’) non-file buffers -- moreover, making them
  ;; buffer local (‘make-variable-buffer-local’) did not help
  (setq xog-quieten-level 0)
  (setq xog-invisible-areas-list ())

  ;; Xumber mode
  ; both work the same
  (autoload ’xumber-mode
    xog-xumber-file
    "Toggle xog number mode."
    t)                                  ; ’t’ for interactive
  (xumber-mode t)
  (if xumber-mode
      (message "  xog: xog-mode: Xumber mode now active")
    (message "  xog: xog-mode: Xumber mode INACTIVE"))

  ;; completion reporting
  (message "  xog: xog-mode setup complete."))

;; ---------------------------------
;;  custom key maps
;; ---------------------------------

(defvar xog-mode-map nil "Local keymap for Xog mode buffers.")
;; CAUTION: must NOT be (make-local-variable ’xog-mode-map)

;; US keyboards
;;
;; ’prior’ is designated PgUp and ’next’ is PgDn --
;; moreover S-<prior> and S-<next> are not normally
;; bound in emacs

(if xog-mode-map
    ()                                  ; protection against double loading

  ;; keyboard shortcuts
  (setq xog-mode-map (make-sparse-keymap))
  (define-key xog-mode-map [(shift            up)] ’xog-move-warn-prior)
  (define-key xog-mode-map [(shift          down)] ’xog-move-warn-next)
  (define-key xog-mode-map [(shift          menu)] ’xog-show-source-context)
  (define-key xog-mode-map [(control        menu)] ’xog-show-header-context)
  (define-key xog-mode-map [(shift         prior)] ’xog-move-source-prior)
  (define-key xog-mode-map [(shift          next)] ’xog-move-source-next)
  (define-key xog-mode-map [(shift          home)] ’xog-move-source-define)
  (define-key xog-mode-map [(shift         right)] ’xog-move-pplus-next)
  (define-key xog-mode-map [(shift          left)] ’xog-move-pplus-prior)
  (define-key xog-mode-map [(shift           end)] ’xog-move-cta-next)
  (define-key xog-mode-map [(shift         pause)] ’xog-quieten-cycle)
  (define-key xog-mode-map [(shift         print)] ’xog-unquieten)

  ;; custom menu (does not make use of the ’easymenu’ elisp package)
  (defvar menuXog0 (make-sparse-keymap "Xog Menu"))    ; mouse label
  (defvar menuXog1 (make-sparse-keymap "Xog1"))
  (defvar menuXog2 (make-sparse-keymap "Xog2"))
  (defvar menuXog3 (make-sparse-keymap "Xog3"))
  (defvar menuXog4 (make-sparse-keymap "Xog4"))

  ;; mouse usage
  (define-key xog-mode-map [C-S-down-mouse-1] menuXog0)

  ;; top-level entry (visible in menu bar, right of "Tools")
  (define-key xog-mode-map [menu-bar xog]          (cons "Xog"             menuXog0)) ; menu-bar label

  ;; main menu bottom (reverse order)
  (define-key menuXog0 [about]       ’(menu-item "About"                xog-menu-about    :help ⬀
"One-line report on Xog mode"))
  (define-key menuXog0 [help]        ’(menu-item "Describe mode"        xog-menu-describe :help ⬀
"Standard emacs ’describe-mode’ call"))
  (define-key menuXog0 [bindings]    ’(menu-item "Describe key bindings" describe-bindings :help ⬀
"Standard emacs ’describe-bindings’ call"))
  (define-key menuXog0 [separator-1] ’("--"))

  ;; sub-menu entries (reverse order)
  (define-key xog-mode-map [menu-bar xog submenu4] (cons "Miscellaneous"   menuXog4))
  (define-key xog-mode-map [menu-bar xog submenu3] (cons "Restrict"        menuXog3))
  (define-key xog-mode-map [menu-bar xog submenu2] (cons "Color"           menuXog2))
  (define-key xog-mode-map [menu-bar xog submenu1] (cons "Navigate"        menuXog1))

  ;; main menu top (reverse order)
  (define-key menuXog0 [separator-2] ’("--"))
  (define-key menuXog0 [tog-screen]  ’(menu-item "Full-screen (toggles)"       ⬀
```

```
xog-fullscreen-me-toggle    :help "Toggle full screen (Linux only)"))
  (define-key menuXog0 [repeat]      '(menu-item "Repeat last command"         repeat                 ⬋
:help "Standard emacs 'repeat' command"))
  (define-key menuXog0 [write-xog]   '(menu-item "Write free buffer to file"   xog-write-xog           ⬋
:help "Write unvisited buffer to file after extracting the filename from the contents of the buffer"))
; (define-key menuXog0 [switch]      '(menu-item "Switch to useful buffer"     ⬋
xog-switch-to-useful-buffer :help "Switch to useful buffer (can use C-x b to return)"))
  (define-key menuXog0 [cycle-bufs]  '(menu-item "Cycle to next useful buffer" xeona-cycle-buffers     ⬋
:help "Cycle to next useful buffer"))
  (define-key menuXog0 [readonly]    '(menu-item "Toggle read-only"            toggle-read-only       ⬋
:help "Standard emacs 'toggle-read-only' command"))
  (define-key menuXog0 [separator-3] '("--"))
  (define-key menuXog0 [header]      '(menu-item "Visit C++ associated header" xog-show-header-context ⬋
:help "Visit C++ header file in read-only mode (C-x k to 'kill-buffer')"))
  (define-key menuXog0 [source]      '(menu-item "Visit C++ source"            xog-show-source-context ⬋
:help "Visit C++ source file in read-only mode and then center on given line (C-x k to ⬋
'kill-buffer')"))

  ;; ---

  ;; submenu 1 (reverse order)
  (define-key menuXog1 [glpk-prior]  '(menu-item "Previous GLPK"           xog-move-glpk-prior    ⬋
:help "Move up to previous GLPK call"))
  (define-key menuXog1 [glpk-next]   '(menu-item "Next GLPK"               xog-move-glpk-next     ⬋
:help "Move down to next GLPK call"))
  (define-key menuXog1 [call-set]    '(menu-item "Set call column regex"   xog-move-call-define   ⬋
:help "Set call column regex for next and previous call commands"))
  (define-key menuXog1 [call-prior]  '(menu-item "Previous call"           xog-move-call-prior    ⬋
:help "Move up to previous manually defined call regex (see set call column regex)"))
  (define-key menuXog1 [call-next]   '(menu-item "Next call"               xog-move-call-next     ⬋
:help "Move down to next manually defined call regex (see set call column regex)"))
  (define-key menuXog1 [src-set]     '(menu-item "Set source column regex" xog-move-source-define ⬋
:help "Set source column regex for next and previous source commands"))
  (define-key menuXog1 [src-prior]   '(menu-item "Previous source"         xog-move-source-prior  ⬋
:help "Move up to previous manually defined source regex (see set source column regex)"))
  (define-key menuXog1 [src-next]    '(menu-item "Next source"             xog-move-source-next   ⬋
:help "Move down to next manually defined source regex (see set source column regex)"))
  (define-key menuXog1 [cta-next]    '(menu-item "Next CTA algorithm"      xog-move-cta-next      ⬋
:help "Move down to next CAPSET or TRANSOLVE marker"))
  (define-key menuXog1 [yeek-prior]  '(menu-item "Previous yeek"           xog-move-yeek-prior    ⬋
:help "Move up to previous yeek"))
  (define-key menuXog1 [yeek-next]   '(menu-item "Next yeek"               xog-move-yeek-next     ⬋
:help "Move down to next yeek"))
  (define-key menuXog1 [pplus-prior] '(menu-item "Previous ++"             xog-move-pplus-prior   ⬋
:help "Move up to previous ++ simulation marker"))
  (define-key menuXog1 [pplus-next]  '(menu-item "Next ++"                 xog-move-pplus-next    ⬋
:help "Move down to next ++ simulation marker"))
  (define-key menuXog1 [warn-prior]  '(menu-item "Previous WARN"           xog-move-warn-prior    ⬋
:help "Move up to previous WARN log"))
  (define-key menuXog1 [warn-next]   '(menu-item "Next WARN"               xog-move-warn-next     ⬋
:help "Move down to next WARN log"))

  ;; submenu 2 (reverse order)
  (define-key menuXog2 [uncolorize]  '(menu-item "Uncolorize buffer"                   hi-lock-mode    ⬋
:help "Remove color emphasis from buffer (by running 'hi-lock-mode')"))
  (define-key menuXog2 [colorize]    '(menu-item "Colorize buffer"                     xog-colorize    ⬋
:help "Add color emphasis to buffer (normally on by default)"))
  (define-key menuXog2 [hi-phrase]   '(menu-item "Highlight a regex"                   ⬋
highlight-phrase                  :help "Highlight regex phrase (by running 'highlight-phrase')"))
  (define-key menuXog2 [highlight]   '(menu-item "Toggle current line highlight (hcl)" ⬋
highlight-current-line-minor-mode :help "Toggle current line highlight (light-brown) (by running ⬋
'highlight-current-line-minor-mode')"))
; (define-key menuXog2 [hl-line]     '(menu-item "Toggle highlight line"               hl-line-mode    ⬋
:help "Toggle current line highlight (green) (by running 'hl-line-mode')"))
; (define-key menuXog2 [highline]    '(menu-item "Toggle highline (hl)"                highline-mode   ⬋
:help "Toggle current line highlight (grey) (by running 'highline-mode')"))

  ;; submenu 3 (reverse order)
  (define-key menuXog3 [unquieten]   '(menu-item "Reveal hidden rank logs" xog-unquieten     :help ⬋
"Reveal all previously hidden log ranks"))
  (define-key menuXog3 [cycle]       '(menu-item "Cycle hide rank logs"    xog-quieten-cycle :help ⬋
"Cycle log ranks in { WARN info dbug xtra adhc }"))

  ;; submenu 4 (reverse order)
  (define-key menuXog4 [font-dec]    '(menu-item "Decrease text size" text-scale-decrease :help ⬋
"Decrease text size using a 'text-scale' function (also <S-down-mouse-1>)"))

 ) ;; final parenthesis

;; ---------------------------------
;;  aliases
```

```
;; ---------------------------------

;; CAUTION: 'defalias' on 2007 GNU Emacs 21.4 has no optional DOCSTRING

;; ---------------------------------
;;  prioritize coloration
;; ---------------------------------

;; prioritize coloration over current line highlighting
;; source: http://www.emacswiki.org/emacs/HiLock
;; status: works as advertised up to about 2000 lines

(defadvice hi-lock-set-pattern (around use-overlays activate)
  "Mode function 'hi-lock-mode' to prioritize coloration over current line highlight."
  (let ((font-lock-fontified nil))
    ad-do-it))

;; ---------------------------------
;;  buffer switch
;; ---------------------------------

;; key binding common with 'xem.el'

(defun xog-switch-to-useful-buffer ()
  "Switch to useful buffer."
  (interactive)
  (switch-to-buffer (other-buffer)))

;; ---------------------------------
;;  menu support
;; ---------------------------------

(defun xog-menu-about ()
  "About message for 'xog-mode'."
  (interactive)
  (let* ((file    (car (load-history-filename-element "xog"))                                ) ; [1]
         (sedstr  "s/^[[:space:]]*;\\+[[:space:]]*\\$Revision: \\([[:digit:]]*\\) \\$/\\1/p" )
         (call    (concat "sed --quiet '" sedstr "' " file)                                 )
         (svnver "(not set)"                                                                ))
    (message "  xog: about: file: '%s'" file)
    (message "  xog: about: sed: '%s'" sedstr)
    (setq svnver (shell-command-to-string call))
    (if (string-match "\n$" svnver)                     ; trim trailing newline
        (setq svnver (replace-match "" t t svnver)))    ; fixed case and literal
    (message "  xog: about: svnver: '%s'" svnver)
    (message "%s major mode %s for reviewing xeona output." mode-name svnver)))
    ;; [1] this file

(defun xog-menu-describe ()
  "Wrapper to 'describe-mode'."
  (interactive)
  (describe-mode))

;; ---------------------------------
;;  full screen mode
;; ---------------------------------

(defun xog-fullscreen-me-toggle ()
  "Toggle full-screen mode."
  (interactive)
  (xog-fullscreen-me t))

(defun xog-fullscreen-me (&optional toggle)
  "Adopt full-screen mode.  If TOGGLE, then alternate with part-screen mode."
  ;; CAUTION: contains platform-specific code
  ;; http://www.emacswiki.org/emacs/FullScreen
  (cond
   ((eq system-type 'gnu/linux)
    ;; 1 = fixed full, 2 = toggle for each call
    (if toggle
        (x-send-client-message nil 0 nil "_NET_WM_STATE" 32 '(2 "_NET_WM_STATE_FULLSCREEN" 0))
      (x-send-client-message nil 0 nil "_NET_WM_STATE" 32 '(1 "_NET_WM_STATE_FULLSCREEN" 0))))
   ((eq system-type 'windows-nt)
    (message "  xog: fullscreen-me: WARNING: full-screen mode not coded for Windows operating system"))
   (t
    (message "  xog: fullscreen-me: WARNING: full-screen mode not coded for this operating system"))) )

;; ---------------------------------
;;  CTA, ++ navigation
;; ---------------------------------
```

```
(defconst xog-move-cta-regex " CAPSET-[0-9][0-9] \\| TRANSOLVE-[0-9][0-9] ")

(defun xog-move-cta-next ()
  "Move to next CTA (capset and transolve algorithm) entry."
  (interactive)
  (let ((regex  xog-move-cta-regex))
    (if (xog-move-regex-next regex)
        (message " xog: move to next ++ using '%s' complete." regex)
      (message "  xog: move to next ++ using '%s' FAILED." regex))))

(defconst xog-move-pplus-regex "^   \\+\\+ ")

(defun xog-move-pplus-next ()
  "Move to next ++ entry."
  (interactive)
  (let ((regex  xog-move-pplus-regex))
    (if (xog-move-regex-next regex)
        (message " xog: move to next ++ using '%s' complete." regex)
      (message "  xog: move to next ++ using '%s' FAILED." regex))))

(defun xog-move-pplus-prior ()
  "Move to prior ++ entry."
  (interactive)
  (let ((regex  xog-move-pplus-regex))
    (if (xog-move-regex-prior regex)
        (message " xog: move to previous plusplus using '%s' complete." regex)
      (message "  xog: move to previous plusplus using '%s' FAILED." regex))))

(defconst xog-move-yeek-regex "yeek[[:space:]]+[[:digit:]]+$")

(defun xog-move-yeek-next ()
  "Move to next yeek 00 message."
  (interactive)
  (let ((regex xog-move-yeek-regex))
    (if (xog-move-regex-next regex)
        (message " xog: move to next yeek using '%s' complete." regex)
      (message "  xog: move to next yeek using '%s' FAILED." regex))))

(defun xog-move-yeek-prior ()
  "Move to previous yeek 00 message."
  (interactive)
  (let ((regex xog-move-yeek-regex))
    (if (xog-move-regex-prior regex)
        (message " xog: move to previous yeek using '%s' complete." regex)
      (message "  xog: move to previous yeek using '%s' FAILED." regex))))

;; ---------------------------------
;;  navigation support
;; ---------------------------------

(defun xog-move-regex-next (regex)
  "Navigate to next REGEX."
  (let ()
    (if (looking-at regex)
        (forward-char +1))                    ; need to "kick-start" the search
    (if (re-search-forward regex nil t)       ; limit (no), no error (true)
        (progn
          (recenter xog-recenter)
          (message "  xog: move-regex-next: move to next '%s' complete." regex)
          t)
      (message "  xog: move-regex-next: move to next '%s' FAILED." regex)
      nil)))

(defun xog-move-regex-prior (regex)
  "Navigate to previous REGEX."
  (let ()
    (if (re-search-backward regex nil t)      ; limit (no), no error (true)
        (progn
          (recenter xog-recenter)
          (message "  xog: move-regex-prior: move to previous '%s' complete." regex)
          t)
      (message "  xog: move-regex-prior: move to previous '%s' FAILED." regex)
      nil)))

;; ---------------------------------
;;  call navigation
;; ---------------------------------

(defvar xog-move-call-regex "" "Call column navigation pattern.")

(defun xog-move-call-define (regex)
```

```
  "Seek and record call column navigation pattern REGEX."
  (interactive "sEnter call column regex: ")
  (let ((was xog-move-call-regex))
    (setq xog-move-call-regex regex)
    (message "  xog: move call define complete moving from '%s' to '%s'." was regex)))

(defun xog-move-call-next ()
  "Navigate to next call log."
  (interactive)
  (let ((regex    xog-move-call-regex)
        (parts    (make-list 7 ""))       )   ; dummy for first iteration
        (start    (point)))
    (message "  xog: move-call-next: %s" parts)
    (while (and (not (eobp))
                (not (string-match regex (nth 2 parts))))   ; (string-match "a+" "") returns 'nil'
      (forward-line +1)
      (setq parts (xog-split-repx (thing-at-point 'line)))
      (if parts () (setq parts (make-list 7 "")) ))
    (recenter xog-recenter)
    (if (eobp)
        (progn
          (goto-char start)
          (message "  xog: move to next call '%s' FAILED." regex))
      (message "  xog: move to next call '%s' complete." regex))))

(defun xog-move-call-prior ()
  "Navigate to previous call log."
  (interactive)
  (let ((regex    xog-move-call-regex)
        (parts    (make-list 7 ""))       )   ; dummy for first iteration
        (start    (point)))
    (message "  xog: move-call-next: %s" parts)
    (while (and (not (bobp))
                (not (string-match regex (nth 2 parts))))   ; (string-match "a+" "") returns 'nil'
      (forward-line -1)
      (setq parts (xog-split-repx (thing-at-point 'line)))
      (if parts () (setq parts (make-list 7 "")) ))
    (recenter xog-recenter)
    (if (bobp)
        (progn
          (goto-char start)
          (message "  xog: move to previous call '%s' FAILED." regex))
      (message "  xog: move to previous call '%s' complete." regex))))

;; ---------------------------------
;;  source navigation
;; ---------------------------------

(defvar xog-move-source-regex "" "Source column navigation pattern.")

(defun xog-move-source-define (regex)
  "Seek and record source column navigation pattern REGEX."
  (interactive "sEnter source column regex (single escape real dots): ")
  (let ((was xog-move-source-regex))
    (setq xog-move-source-regex regex)
    (message "  xog: move source define complete moving from '%s' to '%s'." was regex)))

(defun xog-move-source-next ()
  "Navigate to next source log."
  (interactive)
  (let ((regex    xog-move-source-regex)
        (parts    (make-list 7 ""))       )                 ; dummy for first iteration
        (start    (point)))
    (message "  xog: move-source-next: %s" parts)
    (while (and (not (eobp))
                (not (string-match regex (nth 1 parts))))   ; (string-match "a+" "") returns 'nil'
      (forward-line +1)
      (setq parts (xog-split-repx (thing-at-point 'line)))
      (if parts () (setq parts (make-list 7 "")) ))
    (recenter xog-recenter)
    (if (eobp)
        (progn
          (goto-char start)
          (message "  xog: move to next source '%s' FAILED." regex))
      (message "  xog: move to next source '%s' complete." regex))))

(defun xog-move-source-prior ()
  "Navigate to previous source log."
  (interactive)
  (let ((regex    xog-move-source-regex )
        (parts    (make-list 7 ""))      )                 ; dummy for first iteration
```

```
        (start    (point)               ))
    (message "  xog: move-source-next: %s" parts)
    (while (and (not (bobp))
                (not (string-match regex (nth 1 parts))))   ; (string-match "a+" "") returns 'nil'
      (forward-line -1)
      (setq parts (xog-split-repx (thing-at-point 'line)))
      (if parts () (setq parts (make-list 7 "") )))
    (recenter xog-recenter)
    (if (bobp)
        (progn
          (goto-char start)
          (message "  xog: move to previous source '%s' FAILED." regex))
      (message "  xog: move to previous source '%s' complete." regex))))

;;---------------------------------
;; GLPK navigation
;;---------------------------------

;; CAUTION: the line number below may change
(defconst xog-move-glpk-regex "2735   d/siglp.cc       solverInvokeSolver")

(defun xog-move-glpk-next ()
  "Move to next GLPK message."
  (interactive)
  (let ((regex xog-move-glpk-regex))
    (if (xog-move-regex-next regex)
        (message " xog: move to next GLPK using '%s' complete." regex)
      (message "  xog: move to next GLPK using '%s' FAILED." regex))))

(defun xog-move-glpk-prior ()
  "Move to next GLPK message."
  (interactive)
  (let ((regex xog-move-glpk-regex))
    (if (xog-move-regex-prior regex)
        (message " xog: move to previous GLPK using '%s' complete." regex)
      (message "  xog: move to previous GLPK using '%s' FAILED." regex))))

;; ---------------------------------
;;  WARN navigation
;; ---------------------------------

(defconst xog-warn-tag "WARN")

(defun xog-move-warn-next ()
  "Navigate to next WARN log."
  (interactive)
  (let ((tag    xog-warn-tag)
        (parts  nil          ))
    (while (and (not (eobp))
                (not (string-equal tag (nth 5 parts))))     ; [1]
      (forward-line +1)
      (setq parts (xog-split-repx (thing-at-point 'line))))
    (recenter xog-recenter)
    (if (eobp)
        (message "  xog: move to next WARN FAILED.")
      (message "  xog: move to next WARN complete.")))
      ;; [1] (nth 5 nil) returns nil / (string-equal "abc" nil) returns nil

(defun xog-move-warn-prior ()
  "Navigate to previous WARN log."
  (interactive)
  (let ((tag    xog-warn-tag)
        (parts  nil          ))
    (while (and (not (bobp))
                (not (string-equal tag (nth 5 parts))))
      (forward-line -1)
      (setq parts (xog-split-repx (thing-at-point 'line))))
    (recenter xog-recenter)
    (if (bobp)
        (message "  xog: move to previous WARN FAILED.")
      (message "  xog: move to previous WARN complete."))))

;; ---------------------------------
;;  write lost buffer to XOG file
;; ---------------------------------

(defun xog-write-xog ()
  "Write XOG file after extracting the filename from the contents of the buffer.

Existing files are not backed up, but overwrites are prompted."
  ;; typical "     model file        :  /home/robbie/..../trunk/xeona1/xeona-xmoks/submodel.16.xem"
```

```
  (interactive)
  (save-excursion
    (let ((xog-key        "^     model file        :  " )
          (xog-extension  xog-extension                 )
          (xog-name                                     )
          (capture                                      ))
      (if (buffer-file-name)                ; indicates a visited file
          (message "  xog: write xog encountered a live file '%s' so quiting without action."
                   (abbreviate-file-name buffer-file-name))
        (goto-char (point-min))
        (if (re-search-forward xog-key)
            (progn
              (setq capture (buffer-substring-no-properties (match-end 0) (point-at-eol)))
              (message "  xog: write-xog: points: %d:%d capture: '%s'"
                       (match-end 0) (point-at-eol) capture)
              (setq cature (xog-string-trim capture))
              (setq xog-name (concat (file-name-sans-extension capture) xog-extension))
              (write-file xog-name t)        ; 't' means seek confirmation on overwrite
              (message "  xog: write xog succeeded using '%s'." (abbreviate-file-name xog-name)))
          (message "  xog: write xog cannot find suitable model file (is this really xog output?)."))))
    )))

;; ---------------------------------
;;  visit source code
;; ---------------------------------

(defun xog-show-source-context ()
  "Open the source file indicated by the current log line.
Or else try to fail gracefully."
  (interactive)
  (let ((parts      nil )
        (lineno     0   )
        (source     ""  )
        (filename   ""  )
        (buffername ""  ))
    (setq parts (xog-split-repx (thing-at-point 'line)))
    (if parts
        (progn
          (setq lineno (nth 0 parts))
          (setq lineno (string-to-number lineno))
          (setq source (nth 1 parts))
          (setq filename (file-truename (concat xog-sourcecode source)))
          (if (xog-open-source filename lineno)
              (message "  xog: show source context opening source file '%s' at line '%d'."
                       (substring-no-properties source)     ; remove (yellow) coloration
                       lineno)
            (message "  xog: show-source-context: source file unreadable '%s'" filename)))
      (message "  xog: show source context unable to identify a source file at the current line.")) ))

(defun xog-show-header-context ()
  "Open the associated header file indicated by the current log line.
Or else try to fail gracefully."
  (interactive)
  (let ((parts       nil )
        (lineno      0   )    ; CAUTION: never reset - not a valid lineno but 'emacs' doesn't mind
        (source      ""  )
        (filename    ""  )
        (buffername  ""  )
        (header-ext  "h" ))
    (setq parts (xog-split-repx (thing-at-point 'line)))
    (if parts
        (progn
          (setq source (nth 1 parts))
          (setq source (concat (file-name-sans-extension source) "." header-ext))
          (setq filename (file-truename (concat xog-sourcecode source)))
          (if (xog-open-source filename lineno)
              (message "  xog: show header context opening header file '%s' at line '%d'."
                       (substring-no-properties source)     ; remove (yellow) coloration
                       lineno)
            (message "  xog: show-header-context: header file unreadable '%s'" filename)))
      (message "  xog: show header context unable to identify a source file at the current line.")) ))

(defun xog-open-source (filename lineno)
  "Open FILENAME at LINENO using hardcoded behaviors.

The FILENAME must be absolute.  And LINENO must be an integer.
LINENO zero means use first line.

Currently, the buffer is read-only and the entire current line
is color highlighted."
  (interactive)
```

```
  (if (and (stringp filename)
           (integerp lineno))
      (if (file-readable-p filename)
          (progn
;;;         (find-file filename)             ; also names and switches to buffer
            (find-file-read-only filename)   ; also names and switches to buffer
            (goto-line lineno)
            (recenter nil)
            (highlight-current-line-minor-mode 1)
            (message "  xog: open-source: success"))
        (message "  xog: open-source: filename bad or file not readable %s" filename))
    (message "  xog: open-source: faulty arguments (is lineno an integer)")) )

;; ---------------------------------
;;  repx functions (find and parse logs)
;; ---------------------------------

;; the repx functions recover information made with 'xeona'
;; logging call, which might be invoked by:
;;
;;   s_logger->repx(logga::dbug, "some text", someFloat)
;;
;; function 'xog-split-repx' is used be a number of other
;; functions, including the source code context function and
;; the WARN navigation functions (unless this proves to slow)
;;
;; representative logging output follows
;;
;; line   source           call                    no   delta-t    rank   message                  ⬋
comment/value
;; ⬋
.....................................................................................................⬋
.........................
;; 1869   d/siglp.cc       resetProblem           2150   00.0000s   info   problem reset, rows x cols  ⬋
00 x 00
;;
;; function 'xog-split-repx' is predicated on the fact
;; that separation of at least two spaces is always
;; present -- this is believed to be correct

(defconst xog-rank-regex
  "YEEK\\|KILL\\|WARN\\|info\\|dbug\\|xtra\\|adhc\\|\\?\\?\\?\\?"
  "Repx tags as defined in function 'Logger::calcRank'.")

(defun xog-split-repx (line)
  "Return a list of the eight log elements if LINE is a xeona log line.
Otherwise return nil.

The LINE argument is deemed to have been produced by 'logga::Logger::repx'
after testing its contents.  A false positive is extremely unlikely."
  (interactive)
  (let ((repx-sep    "[ ][ ]+"      )
        (rank-regex  xog-rank-regex ))
    ;; ensure string
    (if (stringp line)
        (progn
          (setq line (xog-string-trim line))
;         (message "  xog: split-repx: line\n    '%s'" line)
          (setq parts (split-string line repx-sep t))
;;;       (if (= (length parts) 7) (push "" parts))    ; this line resulted in a noticeable speedup
;         (message "  xog: split-repx: length %d" (length parts))
;         (dolist (part parts)
;           (message "    %s" (xog-string-trim part)))
;         (message "  xog: split-repx listing complete")
          ;; integrity tests
          ;;
          ;; [1] source file line number __LINE__
          ;; [2] source file name __FILE__
          ;; [3] function name __func__
          ;; [4] repx print number
          ;; [5] elapsed time interval
          ;; [6] repx rank
          ;; [7] message
          ;; [8] can be absent
          (if (and (xog-split-repx-integrity "^[[:digit:]]+$"  (nth 0 parts)  "line")          ; [1]
                   (xog-split-repx-integrity  "\\.h\\|\\.cc$"  (nth 1 parts)  "source")        ; [2]
                   (xog-split-repx-integrity "^[[:print:]]+$"  (nth 2 parts)  "call")          ; [3]
                   (xog-split-repx-integrity "^[[:digit:]]+$"  (nth 3 parts)  "no")            ; [4]
                   (xog-split-repx-integrity  "[[:digit:]]+s$" (nth 4 parts)  "delta-t")       ; [5]
                   (xog-split-repx-integrity  rank-regex       (nth 5 parts)  "rank")          ; [6]
                   (xog-split-repx-integrity "^[[:print:]]+$"  (nth 6 parts)  "message")       ; [7]
```

```
;;;                (xog-split-repx-integrity "^[[:print:]]*$"  (nth 7 parts) "comment/value")  ; [8]
                   ) ; and
              (progn
;               (message "  xog: split-repx: line split and checked for integrity")
                parts)                          ; return meaningful list
;           (message "  xog: split-repx: line split but failed at least one integrity check")
            nil))
;     (message "  xog: split-repx: line not string")
      nil) ))                                   ; return empty list (same as ())

(defun xog-split-repx-integrity (regex input &optional explain)
  "Check if REGEX is able to match INPUT.

If not, provide a readable error message, incorporating EXPLAIN if provided."
  (if (not explain)
      (setq explain "not explained"))
  (if (stringp input)
      (if (string-match regex input)
          t                                     ; return success
        (message "  xog: split-repx-integrity: match with regex \"%s\" and input \"%s\" (%s) failed"
                 regex input explain)
        nil)                                    ; return fail
    (message "  xog: split-repx-integrity: input not string")
    nil))                                       ; return fail

;; ---------------------------------
;;  quieten (progressively hide rank logs)
;; ---------------------------------

;; the rank strings are defined in 'xeona' function 'Logger::calcRank'
;; note also "????" may be used, which regexes to "\\?\\?\\?\\?"

(defconst xog-quieten-ranks
  (list "WARN" "info" "dbug" "xtra" "adhc")  ; add or remove ranks as required
  "Ordered list of log ranks.")

(defvar xog-quieten-level 0
  "Current log rank quieten level.")

(defun xog-quieten-cycle ()
  "Cycle thru levels of quietness."
  (interactive)
  (save-excursion
    (let* ((level   xog-quieten-level  )     ; 24-Jul-2010: can this be an alias?
           (ranks   xog-quieten-ranks  )
           (depth   (length ranks)       )
           (trip    nil                  )
           (count   0                    ))  ; count returned from 'xog-quieten'
      (incf level)                           ; increment level
      (if (> level depth) (setq level 0))    ; roll around to zero as required
      (setq xog-quieten-level level)
      (if (= level 0)                        ; show all
          (progn
            (message "  xog: quieten-cycle: about to call (xog-unquieten)")
            (xog-unquieten)
            (message "  xog: quieten-cycle: reset to level %d." level))
        (setq trip (nth (1- level) (reverse ranks)))   ; simple regex
        (message "  xog: quieten-cycle: trip %s "trip)
        (message "  xog: quieten-cycle: about to call (xog-quieten %s')" trip)
        (setq count (xog-quieten trip))
        (message "  xog: quieten-cycle: level %d with ranks %S removing %d lines this time."
                 level trip count))
      trip)))

;; core hiding code based on 'hide-lines' package by Mark Hulme-Jones
;; source: http://www.emacswiki.org/emacs/hide-lines.el

(defvar xog-invisible-areas-list ()
  "List of invisible overlays used by Xog mode.")

(add-to-invisibility-spec 'xog)

(defun add-invisible-overlay (start end)
  "Add an overlay from START to END for the current buffer.
Push that overlay onto the 'xog-invisible-areas-list' list."
  (let ((overlay (make-overlay start end)))
    (setq xog-invisible-areas-list (cons overlay xog-invisible-areas-list))
    (overlay-put overlay 'invisible 'xog))) ; overlay 'overlay' given property 'invisible' value 'xog'

(defun xog-quieten (ranks)
  "Hide log ranks given by RANKS.
```

```
Use function 'xog-unquieten' to reverse this action.

Some aesthetic problems with current line highlighting may arise,
but these disappear on unquieting."
  (make-variable-buffer-local 'line-move-ignore-invisible)
  (setq line-move-ignore-invisible t)
  (let ((regex  ranks                )
        (parts  nil                  )
        (count  0                    )                ; hide count
        (start                       ))
    (goto-char (point-min))                           ; beginning of buffer
    (while (not (eobp))
      (setq parts (xog-split-repx (thing-at-point 'line)))
      (if (and (not (null parts))                     ; conditions evaluated in order (unlike C)
               (string-match regex (nth 5 parts)))    ; (string-match "a+" "") returns 'nil'
          (progn
            (incf count)
            (beginning-of-line)
            (setq start (point))
            (end-of-line)
            (forward-char +1)
            (add-invisible-overlay start (point)))
        (forward-line +1)))
    (message "  xog: quieten '%s' complete with %d lines removed." regex count)
    count))

(defun xog-unquieten ()
  "Reveal areas previous hidden by 'xog-quieten'."
  (interactive)
  (mapcar (lambda (overlay) (delete-overlay overlay))
          xog-invisible-areas-list)
  (setq xog-invisible-areas-list ())
  (setq xog-quieten-level 0)                  ; reset level
  (xog-colorize)
  (message "  xog: unquieten complete."))

;; ---------------------------------
;;  colorizing
;; ---------------------------------

;; Xll colors  : /etc/X11/rgb.txt
;; show colors : file:///home/robbie/synk/xeona/x11-colors/x11-color-names.html
;; package     : /usr/share/emacs/21.4/lisp/hi-lock.el

;; interesting: aquamarine gold salmon yellow lemon_chiffon

;; complain if not Xll -- defensive programming
(if (eq window-system 'x)                     ; nil = ordinary terminal, 'w32 = MSWin
    (message "  xog: X11 found.")
  (error "  xog: WARNING: X11 assumed but not found"))

;; define colors
(defface xog-hi-color-B02
  '((((background dark)) (:background "gold" :foreground "black"))
    (t (:background "gold")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xog-hi-color-B04
  '((((background dark)) (:background "salmon" :foreground "black"))
    (t (:background "salmon")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xog-hi-color-B06
  '((((background dark)) (:background "burlywood" :foreground "black"))
    (t (:background "burlywood")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xog-hi-color-B07
  '((((background dark)) (:background "beige" :foreground "black"))
    (t (:background "beige")))
  "Face for hi-lock mode."
  :group 'hi-lock-faces)

(defface xog-hi-color-B08
  '((((background dark)) (:background "sandy brown" :foreground "black"))
    (t (:background "sandy brown")))
  "Face for hi-lock mode."
```

```
  :group 'hi-lock-faces)

;; define colorizing regexes
(defconst xog-fake         "this regex should probably never match")
(defconst xog-warn-regex1  "[[:digit:]]s   WARN  ")
(defconst xog-warn-regex2  "^.*[[:digit:]]s   WARN  ")
(defconst xog-BIG_1-regex  " LOOP-[0-9][0-9] \\| OSP-[0-9][0-9] \\| FAC-[0-9][0-9] ")
(defconst xog-BIG_2-regex  " CAPSET-[0-9][0-9] \\| TRANSOLVE-[0-9][0-9] ")
(defconst xog-pplus-regex  " \\+\\+ ")

(defun xog-colorize ()
  "Colorize a xog output buffer."
  (interactive)
  (message "  xog: colorize commencing")
  ;; CAUTION: order is important .. this is like paint
  ;; CAUTION: the first "fake" regex causes any current line
  ;; highlighting to slip gracefully underneath -- tested with
  ;; both 'highlight-current-line-minor-mode' (part of Emacs) and
  ;; 'highline' (not part of Emacs) -- also requires the
  ;; associated 'hi-lock-set-pattern' defadvice
  (highlight-phrase                xog-fake         'xog-hi-color-B04)
; (highlight-lines-matching-regexp xog-warn-regex1  'xog-hi-color-B04)
  (highlight-phrase                xog-warn-regex2  'xog-hi-color-B04)
  (highlight-phrase                xog-BIG_1-regex  'xog-hi-color-B06)
  (highlight-phrase                xog-BIG_2-regex  'xog-hi-color-B06)
  (highlight-phrase                xog-pplus-regex  'xog-hi-color-B06)
  (message "  xog: colorize complete"))

;; ---------------------------------
;;  general utilities
;; ---------------------------------

(defun xog-string-trim (string)
  "Trim STRING from both ends.

Unlike the standard version, this function returns an empty string if supplied a 'nil'."
  ;; [:space:] comprises space, carriage-return, newline, tab, vertical-tab, form-feed
  (let ((regex "^[[:space:]]*\\(.*?\\)[[:space:]]*$")) ; CAUTION: non-greedy '*?' required
    (if (not (stringp string))
        ""                                  ; else could be () to return 'nil'
      (if (string-match regex string)
          (match-string 1 string)           ; 1 indicates first \( \) construct
        string))))                          ; no match, return original

(defun xog-squeeze-lines ()
  "Automatically squeeze all multiple strictly-blank lines in current buffer."
  (interactive)
  (save-excursion                           ; hold cursor position
    (let ((sections   0  )                  ; sections treated
          (lines      0  )                  ; lines processed
          (sect-plural "s" )                ; "section" pluralization
          (line-plural "s" ))               ; "line" pluralization
      (setq lines (count-lines (point-min) (point-max)))              ; original line count
      (goto-char (point-min))               ; start at beginning
      (while (re-search-forward             ; look for matches
              "\n\n\n+"
              nil t)                        ; buffer position bound, return 'nil' not error on fail
        (replace-match
         "\n\n"
         t t)                               ; do not alter case; insert literally
        (setq sections (+ 1 sections)))
      (goto-char (point-min))               ; start at beginning again
      (when (looking-at "\n\n")             ; look for special match
        (kill-line)
        (setq sections (+ 1 sections)))
      (setq lines (- lines (count-lines (point-min) (point-max))))    ; original minus final line count
      (if (= sections 1) (setq sect-plural ""))
      (if (= lines    1) (setq line-plural ""))
      (message "  xog: squeeze-lines complete, processed %d section%s totaling %d line%s."
               sections sect-plural lines line-plural)) ))

;; ---------------------------------
;;  provide
;; ---------------------------------

;; put the mode symbol into the list "features", so that users can
;; invoke (require 'xog-mode) and load your code only when needed

(provide 'xog-mode)

;;; xog.el ends here
```

```
;  $Id: xog.el 5280 2010-10-08 09:28:22Z robbie $
;  end of file
```

```
;;; xumber.el --- extract and manipulate plain text numbers within the editor

;  file-purpose     : emacs extract and manipulate plain text numbers within the editor
;  file-initiator   : Robbie Morrison <robbie@actrix.co.nz>
;  file-create-date : Sat 31-Jul-2010 10:10 UTC
;  file-status      : working
;  file-keywords    : emacs xeona

;  $Revision: 5181 $
;  $Date: 2010-08-24 15:07:53 +0200 (Tue, 24 Aug 2010) $
;  $Author: robbie $
;  $URL: file:///home/robbie/svn-root/xeona/futz/trunk/elisp/xumber.el $

;  XEONA CODEBASE COPY
;
;  * this code originated as part of the xeona codebase
;
;  * the GLPv2 license included here derives from Emacs Lisp
;    coding practice and from not the 'xeona' project --
;    users can optionally chose to adopt the 'xeona' norm
;    instead

;; Copyright (C) 2010 Robbie Morrison

;; This file is *NOT* part of GNU Emacs.

;; This file is free software; you can redistribute it
;; and/or modify it under the terms of the GNU General
;; Public License as published by the Free Software
;; Foundation; either version 2, or (at your option)
;; any later version.

;; This file is distributed in the hope that it will be
;; useful, but WITHOUT ANY WARRANTY; without even the
;; implied warranty of MERCHANTABILITY or FITNESS FOR A
;; PARTICULAR PURPOSE.  See the GNU General Public
;; License for more details.

;; You may have received a copy of the GNU General
;; Public License along with GNU Emacs; see the file
;; COPYING.  If not, write to the Free Software
;; Foundation, Inc., 59 Temple Place - Suite 330,
;; Boston, MA 02111-1307, USA.

;;; Commentary:

;;  ToDos
;;
;;      > add "[+- ]inf" and "nan" searching
;;
;;      * check current buffer logic (although no problems have
;;        been detected), while noting the warning in section 27.2
;;        of the elisp manual
;;
;;      * check `xumber-reformat-number'
;;
;;      * run `checkdoc' periodically
;;
;;  Known issues
;;

;;; History:

;;      * see the subversion log messages

;;; Code:

;; ---------------------------------
;;  user-modifiable global constants
;; ---------------------------------

(defconst xumber-sum-buffer     "*xumber-tally*"   "Buffer for notifying last tally.")
(defconst xumber-history-buffer "*xumber-history*" "Buffer for recording tally history.")

;; ---------------------------------
;;  custom key maps
;; ---------------------------------

(defvar xumber-mode-map nil "Local keymap for Xumber mode buffers.")

(if xumber-mode-map
```

```
    ()                                      ; protection against double loading

  ;; 'super' is the Windows/Penguin key

  ;; keyboard shortcuts
  (setq xumber-mode-map (make-sparse-keymap))
  (define-key xumber-mode-map [(super         right)] 'xumber-show-number)
  (define-key xumber-mode-map [(super          left)] 'xumber-insert-number)
  (define-key xumber-mode-map [(super            up)] 'xumber-move-number-prior)
  (define-key xumber-mode-map [(super          down)] 'xumber-move-number-next)
  (define-key xumber-mode-map [(super        delete)] 'xumber-sum-zero)
  (define-key xumber-mode-map [(super          home)] 'xumber-switch-history-buffer)

  ;; custom menu
  (defvar menuXumber0 (make-sparse-keymap "Xumber Menu"))
  (defvar menuXumber1 (make-sparse-keymap "Xumber1"))

  ;; top-level entry (visible in menu bar, more to the right of "Tools")
  (define-key xumber-mode-map [menu-bar xumber]           (cons "Xumber" menuXumber0))    ; menu-bar ⬋
label

  ;; main menu bottom (reverse order)
  (define-key menuXumber0 [about]       '(menu-item "About"                       xumber-menu-about    ⬋
:help "One-line report on Xumber minor mode (\"s-\" indicates the Windows/Penguin key)"))
  (define-key menuXumber0 [message]     '(menu-item "Open small *Message* buffer" xumber-open-messages ⬋
:help "Open a small message buffer and watch the messages"))
  (define-key menuXumber0 [separator-1] '("--"))

  ;; sub-menu entries (reverse order)
  (define-key xumber-mode-map [menu-bar xumber submenu1]  (cons "Find"   menuXumber1))

  ;; main menu top (reverse order)
  (define-key menuXumber0 [separator-2] '("--"))
  (define-key menuXumber0 [number-1]    '(menu-item "Smart insert formatted number"  ⬋
xumber-insert-number         :help "Insert and reformat prompted number, while respecting the current ⬋
data-type hint in {f i}"))
  (define-key menuXumber0 [reformat]    '(menu-item "Smart reformat current number"  ⬋
xumber-reformat-number       :help "Reformat current number."))
  (define-key menuXumber0 [separator-3] '("--"))
  (define-key menuXumber0 [sum-buffer]  '(menu-item "Switch to tally history buffer" ⬋
xumber-switch-history-buffer :help "Switch to tally history buffer"))
  (define-key menuXumber0 [zero-num]    '(menu-item "Reset number tally"             xumber-sum-zero   ⬋
:help "Reset the number count and tally to zero"))
  (define-key menuXumber0 [find-prior]  '(menu-item "Previous number"                ⬋
xumber-move-number-prior     :help "Find previous number using standard emacs symbol motion and number ⬋
testing"))
  (define-key menuXumber0 [find-next]   '(menu-item "Next number"                    ⬋
xumber-move-number-next      :help "Find next number using standard emacs symbol motion and number ⬋
testing"))
  (define-key menuXumber0 [show-num]    '(menu-item "Show and tally current number"  ⬋
xumber-show-number           :help "Show current number in a potentially more readable format and also ⬋
tally it"))

  ;; ---

  ;; submenu 1 (reverse order)
  (define-key menuXumber1 [find-noeng]  '(menu-item "Find next non-engineering notation"  ⬋
xumber-find-float-noneng  :help "Find next number in scientific notation, but not standard engineering ⬋
notation"))
  (define-key menuXumber1 [find-sci]    '(menu-item "Find next 000.0 scientific notation" ⬋
xumber-find-float-000     :help "Find next number in scientific notation and also using '000.0'"))
  (define-key menuXumber1 [find-dec]    '(menu-item "Find next decimal notation"          ⬋
xumber-find-float-decimal :help "Find next number in decimal notation"))
  (define-key menuXumber1 [find-dec]    '(menu-item "Find next integer notation"          ⬋
xumber-find-integer       :help "Find next number in integer notation"))

  ) ; final parenthesis

;; ---------------------------------
;;  menu support
;; ---------------------------------

(defun xumber-menu-about ()
  "About message for `xumber-mode'."
  (interactive)
  (let* ((file    (car (load-history-filename-element "xumber")) )  ; locate name of this elisp file
         (sedstr  "s/^[[:space:]]*;\\+[[:space:]]*\\$Revision: \\([[:digit:]]*\\) \\$/\\1/p")
         (call    (concat "sed --quiet '" sedstr "' " file)      )
         (svnver  "(not set)"                                    ))
    (message "  xumber: about: file: '%s'" file)
    (message "  xumber: about: sed: '%s'" sedstr)
```

```
    (setq svnver (shell-command-to-string call))
    (if (string-match "\n$" svnver)                    ; trim trailing newline
        (setq svnver (replace-match "" t t svnver)))   ; fixed case and literal
    (message "  xumber: about: svnver: '%s'" svnver)
    (message "Xumber minor mode %s to extract and manipulate plain text /
numbers (\"s-\" indicates the Windows/Penguin key)." svnver)))

;; ---------------------------------
;;  symbol table
;; ---------------------------------

;; modify the syntax table for this mode
;; http://www.gnu.org/software/emacs/manual/html_mono/elisp.html#Syntax-Table-Functions
;; http://www.gnu.org/software/emacs/manual/html_mono/elisp.html#Example-Major-Modes
;; http://www.emacswiki.org/emacs/EmacsSyntaxTable

(defvar xumber-mode-syntax-table
  (let ((st  (make-syntax-table) ))          ; make new syntax table by MODIFYING the existing table
    (modify-syntax-entry ?. "_")             ; put the dot char (.) in class symbol (_)
    ;; the '[' and ']' mean that the number search functions do
    ;; not get hung up on, for example, [999] -- remove the next
    ;; two statements if problems arise because maintaining this
    ;; navigation behavior is not important
    (modify-syntax-entry 91 "_")             ; put '[' in class symbol (_)
    (modify-syntax-entry 93 "_")             ; put ']' in class symbol (_)
    ;; ditto for '(' and ')' for jumping text like (6)
    (modify-syntax-entry 40 "_")             ; put '(' in class symbol (_)
    (modify-syntax-entry 41 "_")             ; put ')' in class symbol (_)
    st)
  "Syntax table used while in `xumber-mode'.")

;; ---------------------------------
;;  development only
;; ---------------------------------

;;; TOFIX: 01-Aug-2010: make message non-centered

(defun xumber-open-messages ()
  "Convenient way of opening *Messages* buffer."
  (interactive)
  (let* ((buffer-name  "*Messages*"                 )
         (w1           (selected-window)            )
         (w2           (split-window-vertically -10) ))
    (other-window 1)
    (switch-to-buffer buffer-name)
    (other-window 1)
    (message "  xumber: open message complete: %S and %S." w1 w2)))

(defun xumber-display-status-buffer (lines)
  "Display and update the status buffer for LINES."
  ;; inspiration: the 'checkdoc' function `checkdoc-display-status-buffer'
  (let ((buffer-tally       xumber-sum-buffer     )
        (buffer-hx          xumber-history-buffer )
        (wmh-prior          window-min-height     )
        (window-min-height  (1+ (length lines))   ))   ; the modeline, if present, counts as one line
    ;; record history buffer
    (setq buffer-hx (get-buffer-create buffer-hx))     ; CAUTION: okay for buffer to exist
;;; (princ (car (last lines)) buffer-hx)
    (princ (car (nth 0 lines)) buffer-hx)              ; zero-based counting
    (princ "\n" buffer-hx)
    ;; display small buffer
    (with-output-to-temp-buffer buffer-tally
      (dolist (line (reverse lines))
        (princ (car line))
        (princ "\n")))
    (message nil)
    (sit-for 0)
    (shrink-window-if-larger-than-buffer          ; CAUTION: statement must be at or near the end
     (get-buffer-window buffer-tally))
    (setq window-min-height wmh-prior) ))

;; ---------------------------------
;;  number navigate functions
;; ---------------------------------

(defun xumber-move-number-next ()
  "Search for next number using symbol motion and the standard `numberp' predicate."
  (interactive)
  (let ((symbol    nil )
        (number    nil )
        (count     0   ))
```

```
    ;; search
    (forward-symbol +1)                     ; nudge
    (while (and (not (eobp))
                (not (number-at-point)))
      (forward-symbol +1)
      (incf count))
    ;; completion reporting
    (if (eobp)
        (message "  xumber: find number reached end of buffer.")
      (setq symbol (thing-at-point 'symbol))
      (setq number (number-at-point))
      (message "  xumber: find number complete with %s and %f" symbol number)) ))

(defun xumber-move-number-prior ()
  "Search for previous number using symbol motion and the standard `numberp' predicate."
  (interactive)
  (let ((symbol    nil )
        (number    nil )
        (count     0   ))
    ;; search
    (forward-symbol -2)                     ; nudge
    (while (and (not (bobp))
                (not (number-at-point)))
      (forward-symbol -1)
      (incf count))
    (forward-symbol +1)
    ;; completion reporting
    (if (bobp)
        (message "  xumber: find number reached start of buffer.")
      (setq symbol (thing-at-point 'symbol))
      (setq number (number-at-point))
      (message "  xumber: find number complete with %s and %f" symbol number)) ))

;; ---------------------------------
;;  number search functions
;; ---------------------------------

(defun xumber-find-float-000 ()
  "Search for -000.0e-00, that is with three leading digits."
  (interactive)
  (let ((regex  "\\<[[:digit:]]\\{3\\}\\.0e[-+][[:digit:]]+\\>" ))
    (if (re-search-forward regex)
        (forward-char -0))                  ; backup 0 chars
    (message "  xumber: find-float complete using regex '%s' and match '%s'."
             regex (match-string 0))))

;; see also 'calculator.el' and (calculator-eng-display)

(defun xumber-find-float-noneng ()
  "Search for -0.0e-02 and similar non-engineering exponents."
  (interactive)
  (let ((regex  "\\<[-+]?[[:digit:]]+\\.e[-+]0[124578]\\|e[-+]1[0134679]\\>" ))
    (if (re-search-forward regex)
        (forward-char -0))
    (message "  xumber: find-float complete using regex '%s' and match '%s'."
             regex (match-string 0))))

(defun xumber-find-float-decimal ()
  "Search for decimal notion with decimal point."
  (interactive)
  (let ((regex  "\\<[-+]?[[:digit:]]+\\.[[:digit:]]+\\>" ))
    (if (re-search-forward regex)
        (forward-char -0))
    (message "  xumber: find-float complete using regex '%s' and match '%s'."
             regex (match-string 0))))

(defun xumber-find-integer ()
  "Search for integer."
  (interactive)
  (let ((regex  "\\<[-+]?[[:digit:]]+\\>" ))
    (if (re-search-forward regex)
        (forward-char 0))
    (message "  xumber: find-integer complete using regex '%s' and match '%s'."
             regex (match-string 0))))

;; ---------------------------------
;;  number display and format functions
;; ---------------------------------

;; CAUTION: setting the sum to "0.0" provokes non-integer status
;; from the outset
```

```
(defvar xumber-number-sum   0 "Xumber tally sum.")
(defvar xumber-number-cnt   0 "Xumber tally count.")

(defun xumber-sum-zero ()
  "Reset tally."
  (interactive)
  (let ((msg1  nil )
        (msgs  ()  ))
    (setq xumber-number-sum  0)
    (setq xumber-number-cnt  0)
    (setq msg1 (format"  xumber: sum-zero (tally %d) : %g" xumber-number-cnt xumber-number-sum))
    (message "%s" msg1)
    (push (list msg1) msgs)
    (xumber-display-status-buffer msgs) ))

(defun xumber-capture-current-number (delete)
  "Attempt to capture current number.  And also delete it if DELETE is true.

Returns a string if something sensible found, otherwise nil."
  ;; NOTE: see commit r4944 for previous "nibble" code which hunted
  ;; for space and newline chars and was platform-specific
  (let ((capture  nil )                      ; type string
        (number   nil )                      ; type number
        (bnds     ()  )                      ; beginning and end positions of symbol
        (beg          )                      ; beginning of symbol
        (end          )                      ; end of symbol
        (size         ))                     ; size of symbol
    ;; CAUTION: requires variable `xumber-mode-syntax-table' which
    ;; puts the dot (.) in the current class symbol (_)
    (setq capture (thing-at-point 'symbol))
    (setq bnds (bounds-of-thing-at-point 'symbol))     ; returns cons cell (START . END)
    (setq beg (car bnds))
    (setq end (cdr bnds))
    (setq size (- end beg))
    (message "  xumber: capture-current-number: size %d" size)
    (setq capture (buffer-substring-no-properties beg end))
    (message "  xumber: capture-current-number: capture '%s'" capture)
    (if (> (length capture) 0)
        (progn
          (setq number (string-to-number capture))
          (message "  xumber: capture-current-number: number %d" number)
          (if delete
              (progn
                (delete-region beg end)
                (message "  xumber: capture-current-number: complete including deletion"))
            (message "  xumber: capture-current-number: complete excluding deletion")))
      (message "  xumber: capture-current-number: complete but capture failed"))
    capture))

(defun xumber-show-number ()
  "Show current number if potentially more readable format."
  ;; comment: could be extended to offer GMk notation
  (interactive)
  (save-excursion
    (let* ((capture  (xumber-capture-current-number nil) )  ; returns a string or null,
           (output   ""                                  )  ; .. 'nil' means do not delete
           (msg      ()                                  )
           (msgs     ()                                  )
           (msg1     nil                                 )
           (msg2     nil                                 ))
      (if capture
          (progn
            (setq number (string-to-number capture))
            (cond
             ;; integer
             ((integerp number)
              (cond
               ;; big integer
               ((> (abs number) 9999)
                (setq output (format "%s = %s"
                                     (format "%i" number)
                                     (xumber-comma-ize number))))
               ;; remaining
               (t
                (setq output (format "%i" number)))))
             ;; float
             (t
              (cond
               ;; big float
               ((> (abs number) 999999.0)
```

```
                (setq output (format "%s = %s"
                                     (format "%e" number)
                                     (xumber-comma-ize number))))
               ;; medium float
               ((> (abs number)      9.0)
                (setq output (format "%s = %s"
                                     (replace-regexp-in-string
                                      "\\.$" ""
                                      (replace-regexp-in-string "0+$" "" (format "%f" number)))
                                     (xumber-comma-ize number))))
               ;; zero
               ((= (abs number)      0.0)
                (setq output "zero"))

               ;; tiny float
               ((< (abs number)      0.0001)
                (setq output (format "%s"
                                     (format "%e" number))))
               ;; small float
               ((< (abs number)      0.1)
                (setq output (format "%s"
                                     (format "%f" number))))
               ;; remaining (0.1% to 100%)
               (t
                (setq output (format "%s (%s)"
                                     (replace-regexp-in-string
                                      "\\.$" ""
                                      (replace-regexp-in-string "0+$" "" (format "%f" number)))
                                     (format "%.0f%%" (* number 100))))))))
           ;; update tally
           (setq xumber-number-sum (+ xumber-number-sum number))
           (setq xumber-number-cnt (1+ xumber-number-cnt))
           ;; show tally
           (setq sum xumber-number-sum)
           (setq cnt xumber-number-cnt)
           (setq msg1 (format "  xumber: show number (comma precise to about e-12) : %s = %s"
                              capture output))
           (if (integerp sum)               ; elisp automatically switches to float when encountered
               (setq msg2 (format "  xumber: show number (tally %02d)                    : %d"
                                  cnt sum))
             (setq msg2 (format "  xumber: show number (tally %02d)                   : %+.2e" cnt
                                sum)))
           (message "%s\n%s" msg1 msg2)          ; duplicate as message
           (push (list msg2) msgs)
           (push (list msg1) msgs)
           (xumber-display-status-buffer msgs))  ; CAUTION: must come late
         ;; completion reporting
         (message "  xumber: show number: nothing sensible to show (check cursor location)."))  )))

(defun xumber-reformat-number ()
  "Capture, reformat, and reinsert current number."
  (interactive)
  (let ((capture    nil )                   ; return from `xumber-capture-current-number'
        (number     nil )                   ; type number
        (size           )                   ; size of number as string
        (reformat       )                   ; return from `xumber-insert-number'
        (recapture      ))                  ; converted return for validation against 'capture'
    (setq capture (xumber-capture-current-number t))
    (if (not (null capture))
        (progn
          (setq size (length capture))
          (setq number (string-to-number capture))
          (setq reformat (xumber-insert-number number))     ; unprompted version of call
          (setq recapture (string-to-number reformat))
          (if (= number recapture)                          ; numerically equal
              (if (string-equal capture reformat)           ; string-wise identical
                  (message "  xumber: reformat number DUPLICATES '%s' as '%s' with original size %d."
                           capture reformat size)
                (message "  xumber: reformat number REFORMATS '%s' as '%s' with original size %d."
                         capture reformat size))
            (message "  xumber: reformat number WARNS ABOUT '%s' as '%s' with original size %d."
                     capture reformat size)))
      (message "  xumber: reformat number quit without action")) ))

(defun xumber-insert-number (number)
  "Ask for and reformat NUMBER according to current XEM input requirements.

Makes use of the type prompt, if present."
  (interactive "nEnter number (any format): ")
  (let ((reformat  "(faulty)" )             ; desired reformatting
        (line                 )             ; current line
```

```
        (leftright            )             ; as split on <>
        (left                 )             ; left part
        (bits                 )             ; as split on multiple blanks
        (type                 )             ; field type prompt in { f, i } or something else
        (temp nil             ))
    ;; confirm space
    (if (not (or (= (char-before) 32)        ; peek backwards and check for space
                 (= (char-after)  32)))
        (error "  xumber: insert-number: refusing to insert in current position"))
    ;; obtain field type prompt
    (setq line (thing-at-point 'line))       ; includes trailing newline
    (setq leftright (split-string line "[<>]"))
    (setq left (pop leftright))              ; left-side
    (setq bits (split-string left))          ; use default separator "[ \f\t\n\r\v]+"
    (setq type (car (last bits)))
    (setq type (downcase type))
    (message "  xumber: insert-number recovered type '%s'" type)
    ;; reformat number
    (cond ((and (equal type "i") (integerp number))                   ; integer
           (setq reformat (format "%d" number)))
          ((and (equal type "f") (zerop number))                      ; zero float
           (setq reformat "0.0"))
          ((and (equal type "f") (integerp number) (< number 100))    ; small "integer" float
           (setq reformat (concat (format "%d" number) ".0")))
          ((equal type "f")                                           ; float
           (setq reformat (format "%+e" number)))                     ; remove "+" to remove leading +
          (t                                                          ; fall-thru
           (setq reformat "")))
    ;; attempt to suppress some zeros
;;; (while (string-match "000" reformat)
;;;   (setq reformat (replace-match "00" t t reformat)))
    (setq temp reformat)
    (while (string-match "00" temp)
      (setq temp (replace-match "0" t t temp)))
    (if (= (string-to-number temp) number)   ; numerically equal
        (setq reformat temp))                ; accept the suppression
    ;; insert
    (if (not (= (char-before) 32)) (insert " "))
    (insert reformat)
    (if (not (= (char-after) 32)) (insert " "))
    ;; completion reporting
    (message "  xumber: insert-number complete using '%f' and '%s'." number reformat)
    reformat))

(defun xumber-comma-ize (number)
  "Comma-ise the (float or integral) NUMBER and return it as a string.

Does not truncate 1.2345e-12, but will with slightly smaller numbers.
Large numbers are not a problem."
  ;; comma-ization code base on: http://www.emacswiki.org/emacs/sunrise-commander.el
  (let* ((buffer  (format "%.16f" (abs number))                       ) ; stringify without sign [1]
         (parts   (split-string buffer "\\." t)                      ) ; split decimal
         (intg    (pop parts)                                        ) ; integral part left of decimal
         (frac    (pop parts)                                        ) ; fractional part right of
         (frac    (if (zerop (string-to-number frac)) nil frac)      ) ; ..  decimal, else null
         (frac    (if frac (replace-regexp-in-string "0+$" "" frac)) ) ; trim trailing zeros
         (sign    (if (< number 0) "-" "")                           ) ; set sign (could also add "+")
         (digits  (reverse (split-string intg "" t))                 )
         (result   nil                                               ))
    (message "\nbuffer : %s\nintg   : %s\nfrac   : %s\nsign   : %s" buffer intg frac sign)
    (dotimes (n (length digits) result)                ; comma-ize code
      (if (and (< 0 n) (zerop (% n 3)))
          (setq result (concat "," result)))
      (setq result (concat (pop digits) result)))
    (if frac (setq result (concat result "." frac)))   ; reinstate fractional part
    (setq result (concat sign result)) ))              ; reinstate sign
    ;; [1] was `number-to-string'

(defun xumber-comma-ize-original (number)
  "Comma-ise the (float or integral) NUMBER and return it as a string."
  ;; source: http://www.emacswiki.org/emacs/sunrise-commander.el
  ;; CAUTION: takes floats but chops the fractional part and fails on some negatives
  (let* ((num     (replace-regexp-in-string "\\..*$" "" (number-to-string number)) )
         (digits  (reverse (split-string num "" t))                                )
         (result   nil                                                             ))
    (dotimes (n (length digits) result)
      (if (and (< 0 n) (zerop (% n 3)))
          (setq result (concat "," result)))
      (setq result (concat (pop digits) result))) ))

(defun xumber-switch-history-buffer ()
```

```
  "Switch to tally history buffer and back."
  (interactive)
  (let ((target  (get-buffer xumber-history-buffer) ))
    (if (not (null target))
        (if (not (eq (current-buffer) target))    ; 'eq' for same lisp object
            (progn
              (switch-to-buffer target)
              (xumber-mode t))                    ; enable this function and associated key map
          (switch-to-buffer (other-buffer)))
      (message "  xumber: switch history buffer cannot switch as buffer '%s' has not been created."
               xumber-history-buffer)) ))

;; ---------------------------------
;;  minor mode
;; ---------------------------------

;; CAUTION: I believe the `xumber-mode-map' definitions
;; need to precede `xumber-mode' definition

;;;###autoload(defvar xumber-mode nil)
(define-minor-mode xumber-mode
  "Minor mode for processing numbers in plain text files.

Xumber mode provides functions to extract and manipulate numbers
in plain text.  Xumber mode is designed to compliment the Xem and
Xeona major modes.

With no argument, this command toggles the mode.  A non-null
prefix argument turns on the mode.  A null prefix argument turns
off the mode."

  ;; common items

  :init-value nil
  :lighter " Xumber"                        ; modeline lighter
  :keymap nil
  :global nil                               ; nil means buffer-local minor mode
  :group 'xumber
  :version nil                              ; emacs version (and not mode version)

  ;; code

  (message "  xumber: xumber-mode: commencing")
  (if xumber-mode
      (message "  xumber: toggle on mode")
    (message "  xumber: toggle off mode"))

  ) ; final parenthesis

;; ---------------------------------
;;  provide
;; ---------------------------------

;; put the mode symbol into the list "features", so that users can
;; invoke (require 'xumber-mode) and load your code only when needed

(provide 'xumber-mode)

;;; xumber.el ends here

;  $Id: xumber.el 5181 2010-08-24 13:07:53Z robbie $
;  end of file
```

final page: xeona.r5314.code0.pdf