EUROPEAN

MIDDLEWARE INITIATIVE

SAGA INFORMATION SYSTEM NAVIGATOR

USERS GUIDE FOR C++ PROGRAMMERS

Document Version:	**1.0.0**
EMI Component Version:	**1.0.3**
Date:	**08.11.2012**

Abstract: This document provides the C++ programmer with the information necessary to get started with the SAGA Information System Navigator API using the gLite adapter

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

Document Log

Issue	Date	Comment

TABLE OF CONTENTS

TABLE OF TABLES

1. INTRODUCTION

1.1. PURPOSE

This document is intended to get people started with SAGA Information System Navigator. It contains instructions on how to build and run an application with reference to the gLite adapter for the Information System Navigator. This guide should be read in conjunction with the API documentation.

1.2. DOCUMENT ORGANISATION

The document starts with an overview of the SAGA Information System Navigator API and the API's classes. Guidance on the required environment variables in Section 3 is followed by descriptions of possible exceptions in Section 4. This is followed by a simple example of how to use the API. Section 6 gives details about the use of the filters that are used by the API to select data.

1.3. DOXYGEN

The doxygen html files for the SAGA C++ API can be found in /usr`/share/doc/glite-saga-adapters-cpp/html/classsaga_1_1isn_1_1entity__data__set.html`

1.4. APPLICATION AREA

This guide is intended for use by middleware developers who's code needs to obtain more detailed data about a service than that available via the SAGA Service Discovery API.

1.5. REFERENCES

Table 1: Table of references

R1	Goodale T. *et al*. (2008). *A Simple API for Grid Applications (SAGA)*. Retrieved July 01, 2009, from Open Grid Forum website: http://www.ogf.org/documents/GFD.90.pdf
R2	Fisher S., Wilson A. and Paventhan A. (2009). *SAGA API Extension: Service Discovery API.* Retrieved July 01, 2009, from Open Grid Forum website: http://www.ogf.org/documents/GFD.144.pdf
R3	Andreozzi S. *et al*. (2007). *GLUE Schema Specification version 1.3*. Retrieved July 01, 2009, from Open Grid Forum website: https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.glue-wg/docman.root.background.specifications/doc14185
R4	Andreozzi S. *et al*. (2009). *GLUE Schema Specification version 2.0*. Retrieved July 01, 2009, from Open Grid Forum website: http://www.ogf.org/documents/GFD.147.pdf

2. OVERVIEW

2.1. SAGA

SAGA is the "Simple API for Grid Applications", a high level, application-oriented API for grid application development. This is an Open Grid Forum (OGF) standard, [R1].

2.2. THE SAGA INFORMATION SYSTEM NAVIGATOR API

The Information System Navigator (ISN) API is an extension to SAGA that has not yet been through the OGF standardisation process. The ISN API provides a mechanism to retrieve data from the information model of a service. The quality of the information returned will depend upon the quality of the data in the back-end system. The gLite adapter for the ISN obtains information from a BDII.

This API has been designed to add value to the `org.ogf.saga.sd` package, [R2]. Having selected a service with the `org.ogf.saga.sd.Discoverer` API, this API provides the means to traverse the information model and retrieve data published about that service. Alternatively it is possible to start with a selected entity type rather than a service. An optional filter can be used to restrict the results returned.

2.3. INFORMATION MODEL

This API can be used to navigate any information system that can be represented as an entity relationship model. The information models supported is dependent on the adapter, with the gLite adapter providing support for GLUE 1 [R3] and GLUE 2 [R4].

2.4. CLASSES

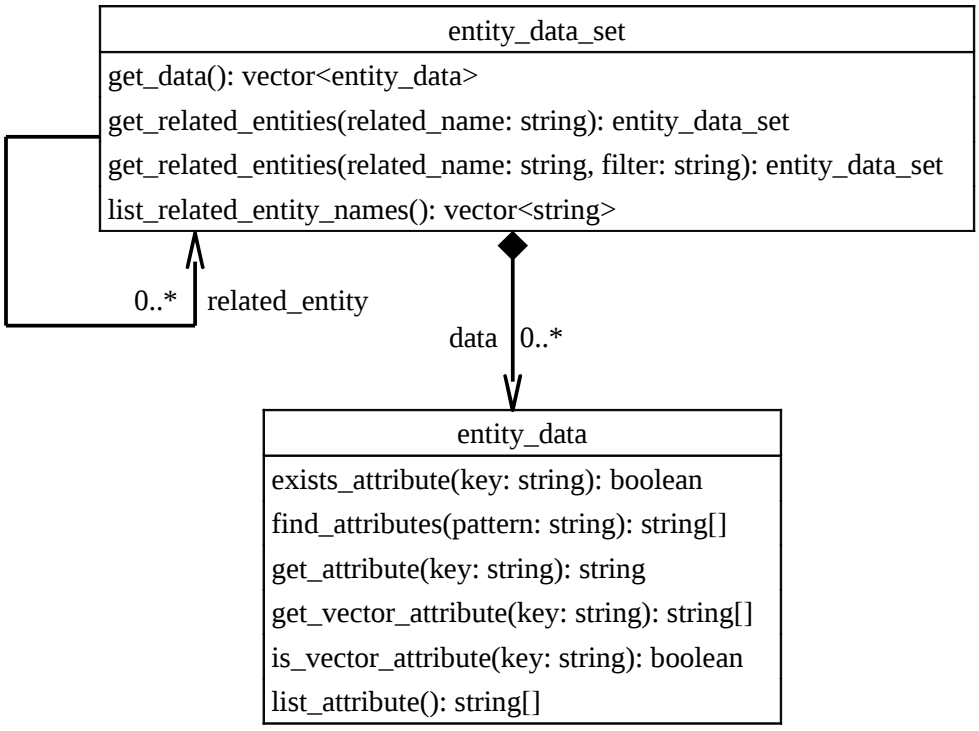

Figure 1. Information System Navigator Entity Relationship Diagram

This API consists of an `entity_data_set` class which contains a set of `entity_data` objects. The `entity_data_set` class has methods `list_related_entity_names` and `get_related_entities`. The `get_data` method returns a set of objects of the `entity_data`

class, with each `entity_data` object representing an instance of an entity as described in the GLUE entity relationship model. The `list_related_entity_names` method returns a list of names of entities for use with the `get_related_entities` method, with the names represent the entities, in the GLUE entity relationship model, that can be navigated to from the current entity. The `get_related_entities` method returns an object of the `entity_data_set` class, filtered according to a specified filter, Section 6.

The `entity_data` gives *ReadOnly* access to all the key names and values in the `entity_data` object.

2.5. GLITE ADAPTER

The gLite adapter uses the BDII information system. It may try to use a BDII but not be able to access it. If no result can be returned because of BDII or other internal problems, it will throw the `NoSuccess` exception.

The adapter provides support for the GLUE 1 and GLUE 2. This is done via a series of XML files provided by the `org.glite.saga-adapter.isn-common` component. When using the API the information model has to be specified, the gLite adapter currently supports the values 'glue1' and 'glue2', see Section 5.

3. GETTING STARTED WITH SAGA INFORMATION SERVICE NAVIGATOR

3.1. INSTALLATION

It is recommended that yum is used to install the client:

```
yum install emi.saga-adapter.isn-cpp
```

3.2. ENVIRONMENT VARIABLES

You should ensure the required variables are set as listed below.

3.2.1. Building

`SAGA_LOCATION=/usr`

The SAGA_LOCATION is the location of the C++ installation of SAGA.

Boost headers to match the installed boost libraries are needed (typically `boost-devel` in the case of an RPM).

3.2.2. Executing

`SAGA_LOCATION=/usr`

The SAGA_LOCATION is the location of the C++ installation of SAGA.

`LD_LIBRARY_PATH=$SAGA_LOCATION/lib/`

The LD_LIBRARY_PATH is required by the C++ adapter.

`BDII_URL=ldap://host.domain:2170`

The BDII_URL is the URL of the LDAP server to be used by the adapter when a URL is not passed in via the API. This takes preference over the LCG_GFAL_INFOSYS and the hard coded value of the CERN bdii.

`LCG_GFAL_INFOSYS=host.domain:2170`

LCG_GFAL_INFOSYS will be used if BDII_URL is not set. This takes preference over the hard coded value of the CERN bdii.

4. SAGA EXCEPTIONS

SagaException is the base class for all exceptions in SAGA.

AuthenticationFailedException is thrown if none of the available session contexts could successfully be used for authentication.

AuthorizationFailedException is thrown if none of the available contexts of the used session could be used for successful authorization. This error indicates that the resource could not be accessed at all, and not that an operation was not available due to restricted permissions.

BadParameterException is thrown if a filter has an invalid syntax or if a filter uses invalid keys.

DoesNotExistException is thrown if the URL is syntactically valid, but no service can be contacted at that URL.

IncorrectState is thrown if the object a method was called on is in a state where that method cannot possibly succeed.

IncorrectURLException is thrown if an implementation cannot handle the specified protocol, or that access to the specified entity via the given protocol is impossible.

NoSuccessException is thrown if no result can be returned because of information system or other internal problems.

NotImplementedException is thrown if the method is not implemented by that SAGA implementation at all.

PermissionDenied is thrown when the method failed because the identity used did not have sufficient permissions to perform the operation successfully.

TimeoutException is thrown if a remote operation did not complete successfully because the network communication or the remote service timed out.

5. A SIMPLE USE CASE AND EXAMPLE

5.1. USE CASE

A use case, which is illustrated below 5.2, is where data regarding selected sites are required. For this example we use the information model “glue1” and select the entity “Site”. To restrict the sites returned by the query the filter “Description='LCG Site'” is used, where “Description” is an attribute of the “Site” entity. An `EntityDataSet` object is returned in response to the query. This object contains a set of `EntityData`, with each `EntityData` relating to details about an individual site. Section 5.6 shows how to extract the data for each site.

5.2. THE CODE

```
1 #include "saga/saga/isn.hpp"

2 int main(int argc, char *argv[])
3 {
4   std::string model = "glue1";
5   std::string entity_name = "Site";
6   std::string filter = "Description='LCG Site'";

7   try
8   {
9     // Create an EntityDataSet
10    saga::isn::entity_data_set eds(model, entity_name, filter);

11    std::cout << "Selected " << eds.get_entity_count() << " sites"
12      << std::endl;

13    std::vector<std::string> rel = eds.list_related_entity_names();
14    std::vector<std::string>::const_iterator iter;
15    std::vector<std::string>::const_iterator endIter = rel.end();

16    std::cout << "Related Enities:"<< std::endl;

17    for ( iter = rel.begin(); iter != endIter; ++iter )
18    {
19      std::cout << "    " << *iter << std::endl;
20    }
21  }

22  catch ( saga::exception& e )
23  {
24    std::cerr << "ERROR: " << e.get_message() << std::endl;
25    exit(1);
```

```
26   }

27   return 0;
28 }
```

5.3. EXPLANATION OF THE CODE

Line 1 is the include statement.

Line 4 sets the name of the data model to be used. This can be 'glue1' or 'glue2'.

Line 5 sets the name of the entity to be selected. This has to be an entity defined in the selected data model.

Line 6 specifies a filter to be used when populating the entities data set. The use of filters is described in Section 6.

Line 10 creates an entity_data_set. The underlying adapter will attempt to use the value set in the environment variable BDII_URL as the URL to contact the BDII, see 3.2.2.

Lines 11-12 displays the count of entities (sites) returned.

Lines 13-20 displays the names of the entities that can be navigated to from the current entity.

Lines 22-26 reports any SAGA exceptions.

5.4. BUILDING THE EXAMPLE

Ensure that the environment is setup correctly (see 3.2). The following makefile will build the example.

```
NUM = one
SRC = isn_example_$(NUM).cpp
OBJ = $(SRC:%.cpp=%.o)
BIN = isn_example_$(NUM)

CXXFLAGS =
LDFLAGS = -l saga_package_isn

all: $(BIN)

$(BIN): $(OBJ)
	$(CXX) $(LDFLAGS) -o $@ $<

clean:
	@$(RM) $(OBJ) $(BIN)
```

Assuming the above is in a file called Makefile, the command 'make' will build the example

5.5. RUNNING THE EXAMPLE

Ensure that the environment is set up correctly (see 3.2). Assuming you are in the directory where sd_example_one is then

```
./isn_example_one
```

will run the test application.

5.6. ACCESSING THE DATA FIELDS

In order to examine the contents of the data associated with an entity add the following after line 21:

```
1     // Extract the data set
2     std::vector<saga::isn::entity_data> data_set = eds.get_data();

3     std::vector<saga::isn::entity_data>::const_iterator dataIter;
4     std::vector<saga::isn::entity_data>::const_iterator
5     endIter = data_set.end();

6     for ( dataIter = data_set.begin(); dataIter != endIter; ++dataIter )
7     {
8       std::vector<std::string> attribNames = dataIter->list_attributes();

9       std::vector<std::string>::const_iterator attribNamesIter;
10      std::vector<std::string>::const_iterator
11      attribNamesEnd = attribNames.end();

12      for ( attribNamesIter = attribNames.begin();
13          attribNamesIter != attribNamesEnd;
14          ++attribNamesIter )
15      {
16        if ( !dataIter->attribute_is_vector(*attribNamesIter) )
17        {
18          std::string
19          attribValue = dataIter->get_attribute(*attribNamesIter);
20          std::cout << *attribNamesIter << ": "
21          << attribValue << std::endl;
22        }

23        else
24        {
25          std::vector<std::string> attribValues =
26          dataIter->get_vector_attribute(*attribNamesIter);

27          std::vector<std::string>::const_iterator attribValuesIter;
28          std::vector<std::string>::const_iterator
29          attribValuesEnd = attribValues.end();

30          for ( attribValuesIter = attribValues.begin();
```

```
31              attribValuesIter != attribValuesEnd;
32              ++attribValuesIter )
33          {
34            std::cout << *attribNamesIter << ": "
35            << *attribValuesIter << std::endl;
36          }
37        }
38      }
39      std::cout << std::endl;
40    }
```

Line 2 extracts the data set.

Lines 6-40 loop round the data set.

Lines 12-38 loop round the attributes of an individual entry in the data set and displays them. Scalar and vector attributes have to be handled separately.

5.7. ACCESSING RELATED ENTITIES

To navigate to a different entity, for example from the “Site” entity to the “Cluster” entity add the following line of code:

```
saga::isn::entity_data_set
      clusterEntity = eds.get_related_entities("Cluster");
```

This creates a new entity_data_set that contains data about all “Clusters” corresponding to the originally selected “Sites”. N.B. the new set will contain entries for every cluster at each of the originally selected sites.

In order to restrict the number of cluster selected we can us a filter with the get_related_entities method. For example:

```
saga::isn::entity_data_set
      clusterEntity = eds.get_related_entities("Cluster","Name='xxxx'");
```

The filter can be used to evaluate one or more of the clusters attributes, see Section 6 for a description of the use of filters.

As with the data about each site, the data associated with each cluster can be examined, as described in 5.6.

6. FILTER

There is a filter string which acts to restrict the set of entities returned. The filter string uses SQL92 syntax as if it were part of a `WHERE` clause acting to select from a single table. SQL92 has been chosen because it is widely known and has the desired expressive power. Multi-valued attributes are treated as a set of values.

Only the following operators are permitted in expressions not involving multi-valued attributes: `IN`, `LIKE`, `AND`, `OR`, `NOT`, `=`, `>=`, `>`, `<=`, `<`, `<>` in addition to column names, parentheses, column values as single quoted strings, numeric values and the comma. For a multi-valued attribute, the name of the attribute MUST have the keyword `ALL` or `ANY` immediately before it, unless comparison with a set literal is intended. For each part of the expression, the attribute name MUST precede the literal value.

The `LIKE` operator matches string patterns:

- `'%xyz'` matches all entries with trailing xyz
- `'xyz%'` matches all entries with leading xyz
- `'%xyz%'` matches all entries with xyz being a substring

The `ESCAPE` keyword can be used with `LIKE` in the normal way.

Column names and values are case sensitive .

The operators `>=`, `>`, `<= and <` are not supported when applied to strings.

Column names in the the filter string are matched against the attribute names of the entity.

If values are specified as numeric values and not in single quotes, the values will be converted from string to numeric for comparison.

Attributes may be multi-valued. If a filter string does not have the correct syntax to accept multi-valued attributes, and an entity has more than one value for an attribute mentioned in the filter, that entity MUST be rejected.

Some examples are:

```
Description='LCG Site' OR Description='EGEE Site'
Latitude >= 0 AND Latitude <= 5
```

7. LIMITATIONS

This section lists the limitations of the adapter.

- The use of the keywords UP and DOWN, for use when navigating to related entities, is not implemented.