

DMR

2.0.0

Generated by Doxygen 1.14.0

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 include/dmr.h File Reference	3
2.1.1 Detailed Description	5
2.1.2 Macro Definition Documentation	6
2.1.2.1 CUSTOM_SLURM_BIN_PREFIX	6
2.1.2.2 DMR_AUTO	6
2.1.2.3 DMR_BLOCKING_REQ	6
2.1.2.4 DMR_CHECKPOINT_RESTART	7
2.1.2.5 DMR_DEBUG_LEVEL	7
2.1.2.6 DMR_DEFAULT_INHIBITOR	7
2.1.2.7 DMR_DEFAULT_POLICY_MAX	7
2.1.2.8 DMR_DEFAULT_POLICY_MIN	8
2.1.2.9 DMR_DEFAULT_POLICY_PREF	8
2.1.2.10 DMR_DEFAULT_POLICY_STRIDE	8
2.1.2.11 DMR_JOBS_CAN_GROW	8
2.1.2.12 DMR_JOBS_CAN_SHRINK	8
2.1.2.13 DMR_NODES_IN_EXPAND	9
2.1.2.14 DMR_NODES_IN_SHRINK	9
2.1.2.15 DMR_PRINT_ANALYTICS	9
2.1.2.16 DMR_PROCS_PER_NODE	10
2.1.2.17 DMR_SKIP_SSH_CHECK	10
2.1.2.18 DMR_TALP_SENSITIVITY	10
2.1.2.19 DMR_TALP_TARGET_CE	10
2.1.3 Typedef Documentation	11
2.1.3.1 DMRAction	11
2.1.3.2 DMRSuggestion	11
2.1.4 Enumeration Type Documentation	11
2.1.4.1 DMRActionEnum	11
2.1.4.2 DMRSuggestionEnum	11
2.1.5 Function Documentation	12
2.1.5.1 dmr_cancel_expansion()	12
2.1.5.2 dmr_check()	12
2.1.5.3 dmr_finalize()	13
2.1.5.4 dmr_get_active_expansions()	14
2.1.5.5 dmr_get_current_node_count()	14
2.1.5.6 dmr_get_last_action()	14
2.1.5.7 dmr_get_nodes_next_expand()	14
2.1.5.8 dmr_get_nodes_next_shrink()	15
2.1.5.9 dmr_get_procs_next_expand()	15

2.1.5.10 dmr_get_procs_next_shrink()	15
2.1.5.11 dmr_get_reconfig_count()	16
2.1.5.12 dmr_init()	16
2.1.5.13 dmr_intercomm_available()	17
2.1.5.14 dmr_pending_expansion()	17
2.1.5.15 dmr_reconfigure()	17
2.1.5.16 dmr_set_jobs_next_shrink()	18
2.1.5.17 dmr_set_nodes_next_expand()	18
2.1.5.18 dmr_set_nodes_next_shrink()	18
2.1.5.19 dmr_set_policy_max_nodes()	19
2.1.5.20 dmr_set_policy_min_nodes()	19
2.1.5.21 dmr_set_policy_pref_nodes()	20
2.1.5.22 dmr_set_policy_stride()	20
2.1.5.23 dmr_set_ppn_next_expand()	20
2.1.5.24 dmr_set_procs_next_expand()	21
2.1.5.25 dmr_set_procs_next_shrink()	21
2.1.5.26 dmr_set_reconf_step_inhibitor()	22
2.1.6 Variable Documentation	22
2.1.6.1 DMR_INTERCOMM	22
2.2 src/dmr.c File Reference	22
2.2.1 Detailed Description	24
2.2.2 Function Documentation	24
2.2.2.1 dmr_cancel_expansion()	24
2.2.2.2 dmr_check()	24
2.2.2.3 dmr_finalize()	25
2.2.2.4 dmr_get_active_expansions()	26
2.2.2.5 dmr_get_current_node_count()	26
2.2.2.6 dmr_get_last_action()	26
2.2.2.7 dmr_get_nodes_next_expand()	26
2.2.2.8 dmr_get_nodes_next_shrink()	27
2.2.2.9 dmr_get_procs_next_expand()	27
2.2.2.10 dmr_get_procs_next_shrink()	27
2.2.2.11 dmr_get_reconfig_count()	28
2.2.2.12 dmr_init()	28
2.2.2.13 dmr_intercomm_available()	29
2.2.2.14 dmr_pending_expansion()	29
2.2.2.15 dmr_reconfigure()	29
2.2.2.16 dmr_set_jobs_next_shrink()	30
2.2.2.17 dmr_set_nodes_next_expand()	30
2.2.2.18 dmr_set_nodes_next_shrink()	30
2.2.2.19 dmr_set_policy_max_nodes()	31
2.2.2.20 dmr_set_policy_min_nodes()	31

2.2.2.21 dmr_set_policy_pref_nodes()	32
2.2.2.22 dmr_set_policy_stride()	32
2.2.2.23 dmr_set_ppn_next_expand()	32
2.2.2.24 dmr_set_procs_next_expand()	33
2.2.2.25 dmr_set_procs_next_shrink()	33
2.2.2.26 dmr_set_reconf_step_inhibitor()	34
2.2.3 Variable Documentation	34
2.2.3.1 DMR_INTERCOMM	34
Index	35

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

include/ dmr.h	
DMR main include file	3
src/ dmr.c	
DMR main functionality	22

Chapter 2

File Documentation

2.1 include/dmr.h File Reference

DMR main include file.

```
#include <mpi.h>
```

Macros

- `#define DMR_DEBUG_LEVEL 0`
Set this variable to display debug prints.
- `#define DMR_PRINT_ANALYTICS 0`
Whether or not to print analytics information.
- `#define DMR_NODES_IN_EXPAND 1`
The default number of nodes to add in each expand.
- `#define DMR_PROCS_PER_NODE 1`
Default number of processes per node in expander jobs.
- `#define DMR_NODES_IN_SHRINK 1`
Default number of nodes to shrink by, if this functionality is available.
- `#define DMR_BLOCKING_REQ 0`
Whether or not to block while waiting for resources.
- `#define DMR_CHECKPOINT_RESTART 1`
Whether or not to use checkpoint-restart for shrinking.
- `#define DMR_JOBS_CAN_SHRINK 1`
Whether or not we can manually adjust the sizes of jobs to shrink.
- `#define DMR_JOBS_CAN_GROW 0`
Whether or not we are able to use the Slurm feature of expanding running jobs.
- `#define DMR_TALP_SENSITIVITY 15`
Sensitivity factor of DMR's performance-aware policy.
- `#define DMR_TALP_TARGET_CE 0.8`
The target communication efficiency to aim for in the performance-aware policy.
- `#define DMR_DEFAULT_POLICY_MIN 1`
Default minimum node count for policy-based reconfiguration decisions.
- `#define DMR_DEFAULT_POLICY_MAX 1`

- *Default minimum node count for policy-based reconfiguration decisions.*
• #define [DMR_DEFAULT_POLICY_STRIDE](#) 2
- *Default multiplier for policies which involve multiplying the node count.*
• #define [DMR_DEFAULT_POLICY_PREF](#) 1
- *Default preferred node count for policies which respect a preference.*
• #define [DMR_DEFAULT_INHIBITOR](#) 0
- *Default number of iterations to inhibit.*
• #define [CUSTOM_SLURM_BIN_PREFIX](#) ""
- *Path to find Slurm executables.*
• #define [DMR_SKIP_SSH_CHECK](#) 0
- *Whether or not to skip expander job ssh health check.*
• #define [DMR_AUTO](#)(the_action, redist_func, restart_func, finalize_func)
- *Macro to automatically deal with a [DMRAction](#) returned by a DMR function.*

Typedefs

- typedef enum [DMRSuggestionEnum](#) [DMRSuggestion](#)
Used by library user to suggest to DMR which action to take.
- typedef enum [DMRActionEnum](#) [DMRAction](#)
Used by DMR to tell user which action to take on current process.

Enumerations

- enum [DMRSuggestionEnum](#) {
[ROUND_POLICY](#) , [CE_POLICY](#) , [SLURM4DMR_ROUND_POLICY](#) , [SLURM4DMR_CE_POLICY](#) ,
[SLURM4DMR_QUEUE_POLICY](#) , [SHOULD_EXPAND](#) , [SHOULD_SHRINK](#) , [SHOULD_STAY](#) }
Used by library user to suggest to DMR which action to take.
- enum [DMRActionEnum](#) {
[DMR_NO_ACTION](#) , [DMR_RECONF](#) , [DMR_RESTART_RECONF](#) , [DMR_REDIST_FINALIZE](#) ,
[DMR_FINALIZE](#) , [DMR_CLEANUP](#) , [DMR_ERROR](#) }
Used by DMR to tell user which action to take on current process.

Functions

- [DMRAction](#) [dmr_init](#) (int argc, char *argv[])
Initialize the DMR library, unlocking other functionality.
- [DMRAction](#) [dmr_check](#) ([DMRSuggestion](#) suggested_reconfiguration)
Suggest or check on DMR reconfiguration progress.
- [DMRAction](#) [dmr_reconfigure](#) ()
Reconfigure the DMR environment.
- [DMRAction](#) [dmr_finalize](#) ()
Finalize the DMR environment.
- int [dmr_intercomm_available](#) ()
Check whether DMR is currently exposing a usable intercommunicator.
- int [dmr_get_active_expansions](#) ()
Get the number of expansion jobs that are currently active.
- int [dmr_get_reconfig_count](#) ()
Check how many times DMR has reconfigured itself since original launch.
- int [dmr_get_nodes_next_expand](#) ()

- Get the number of nodes to request next time expanding.*

 - void [dmr_set_nodes_next_expand](#) (int nodes)

Set the number of nodes to request next time expanding.
- int [dmr_get_procs_next_expand](#) ()

Get the number of processes to spawn next time expanding.
- void [dmr_set_procs_next_expand](#) (int procs)

Set the EXACT number of processes to spawn next time expanding.
- void [dmr_set_ppn_next_expand](#) (int ppn)

Set the number of processes per node to spawn next time expanding.
- int [dmr_get_nodes_next_shrink](#) ()

Get the number of nodes that would be removed if shrinking.
- void [dmr_set_nodes_next_shrink](#) (int nodes)

Set the number of nodes to remove next time shrinking.
- void [dmr_set_jobs_next_shrink](#) (int jobs)

Set the number of jobs to remove next time shrinking.
- int [dmr_get_procs_next_shrink](#) ()

Get the number of processes that would be removed if shrinking.
- void [dmr_set_procs_next_shrink](#) (int procs)

Set the number of processes to remove next time shrinking.
- int [dmr_get_current_node_count](#) ()

Get the number of nodes that are participating in the current DMR session.
- int [dmr_pending_expansion](#) ()

Indicate whether an expansion job is currently pending.
- void [dmr_cancel_expansion](#) ()

Cancel a pending expansion job.
- void [dmr_set_reconf_step_inhibitor](#) (int steps)

Set an inhibitor for requesting reconfigurations.
- void [dmr_set_policy_min_nodes](#) (int nodes)

Set the minimum nodes when removing nodes by policy.
- void [dmr_set_policy_max_nodes](#) (int nodes)

Set the maximum nodes when adding nodes by policy.
- void [dmr_set_policy_stride](#) (int multiplier)

Set the stride for applicable policies to use.
- void [dmr_set_policy_pref_nodes](#) (int nodes)

Set the preferred number of nodes for applicable policies to use.
- [DMRAction dmr_get_last_action](#) (void)

Get the last action that was returned by DMR.

Variables

- MPI_Comm [DMR_INTERCOMM](#)

The intercommunicator used for data reconfiguration.

2.1.1 Detailed Description

DMR main include file.

Contains the main functionality of DMR to call when using the library Include this file to use the library's functionality.

2.1.2 Macro Definition Documentation

2.1.2.1 CUSTOM_SLURM_BIN_PREFIX

```
#define CUSTOM_SLURM_BIN_PREFIX ""
```

Path to find Slurm executables.

Custom path for expander jobs to look for the scontrol executable. Is needed if the required option is not on PATH.

2.1.2.2 DMR_AUTO

```
#define DMR_AUTO(  
    the_action,  
    redist_func,  
    restart_func,  
    finalize_func)
```

Macro to automatically deal with a [DMRAction](#) returned by a DMR function.

Simplifies the use of DMR through pre-provided responses to the possible actions that are returned by DMR.

Warning

Possible program termination point

Precondition

DMR has not already been finalized with [dmr_finalize\(\)](#)

Parameters

in	<i>the_action</i>	A DMRAction returned by a DMR function
in	<i>redist_func</i>	The function to call to checkpoint or transfer data, or NULL if none needed at the point called
in	<i>restart_func</i>	The function to call to read or receive data when restarting, or NULL if none needed at the point called
in	<i>finalize_func</i>	The function to call to clean up resources, or NULL if none needed at the point called

2.1.2.3 DMR_BLOCKING_REQ

```
#define DMR_BLOCKING_REQ 0
```

Whether or not to block while waiting for resources.

If DMR_BLOCKING_REQ is enabled, [dmr_check\(\)](#) will run indefinitely until any requested resources have been acquired (or the acquisition failed). This may be useful if you are running in an environment in which the resources are very likely to be available, such as Slurm4DMR.

2.1.2.4 DMR_CHECKPOINT_RESTART

```
#define DMR_CHECKPOINT_RESTART 1
```

Whether or not to use checkpoint-restart for shrinking.

Without checkpoint-restart, a custom version of PR RTE is needed which supports the removal of daemons. Alternatively, a non-aggressive Slurm configuration which does not kill the SSH connection to nodes that leave the allocation will also work.

2.1.2.5 DMR_DEBUG_LEVEL

```
#define DMR_DEBUG_LEVEL 0
```

Set this variable to display debug prints.

Level 0 or below: never print debug statements Level 1: print debug statements only if we can determine that they emanate from MPI process 0 Level 2 or above: always print debug statements

Other levels: never print debug statements

Note

You can set an environment variable of the same name when running to override the compiled value

2.1.2.6 DMR_DEFAULT_INHIBITOR

```
#define DMR_DEFAULT_INHIBITOR 0
```

Default number of iterations to inhibit.

Set the default value of the inhibitor that will prevent DMR from requesting any new reconfiguration when `dmr_check` is called. If the inhibitor is `N`, then `N` out of every `N+1` calls to `dmr_check(...)` are ignored. The inhibitor has no effect if a reconfiguration is already underway.

The value can be overridden in the environment or at runtime. The runtime has highest priority, followed by the environment, and finally the compile-time default.

2.1.2.7 DMR_DEFAULT_POLICY_MAX

```
#define DMR_DEFAULT_POLICY_MAX 1
```

Default minimum node count for policy-based reconfiguration decisions.

Used by any DMR policy which makes automated re-sizing decisions. DMR will not reconfigure above this value. The value can be overridden in the environment or at runtime. The runtime has highest priority, followed by the environment, and finally the compile-time default.

2.1.2.8 DMR_DEFAULT_POLICY_MIN

```
#define DMR_DEFAULT_POLICY_MIN 1
```

Default minimum node count for policy-based reconfiguration decisions.

Used by any DMR policy which makes automated re-sizing decisions. DMR will not reconfigure below this value. The value can be overridden in the environment or at runtime. The runtime has highest priority, followed by the environment, and finally the compile-time default.

2.1.2.9 DMR_DEFAULT_POLICY_PREF

```
#define DMR_DEFAULT_POLICY_PREF 1
```

Default preferred node count for policies which respect a preference.

Used by any DMR policy which makes automated re-sizing decisions considering a set user preference. The value can be overridden in the environment or at runtime. The runtime has highest priority, followed by the environment, and finally the compile-time default.

2.1.2.10 DMR_DEFAULT_POLICY_STRIDE

```
#define DMR_DEFAULT_POLICY_STRIDE 2
```

Default multiplier for policies which involve multiplying the node count.

Used by any DMR policy which makes automated re-sizing decisions by multiplying or dividing the current node count. The value can be overridden in the environment or at runtime. The runtime has highest priority, followed by the environment, and finally the compile-time default.

2.1.2.11 DMR_JOBS_CAN_GROW

```
#define DMR_JOBS_CAN_GROW 0
```

Whether or not we are able to use the Slurm feature of expanding running jobs.

Indicates whether or not we can submit an expander job which can be merged into the main job. This requires legacy Slurm functionality to work. Requires job shrinking functionality to be enabled.

2.1.2.12 DMR_JOBS_CAN_SHRINK

```
#define DMR_JOBS_CAN_SHRINK 1
```

Whether or not we can manually adjust the sizes of jobs to shrink.

Whether or not we have access to the Slurm feature of removing nodes from jobs while they are running.

2.1.2.13 DMR_NODES_IN_EXPAND

```
#define DMR_NODES_IN_EXPAND 1
```

The default number of nodes to add in each expand.

How many nodes to add each time an expansion is requested by the DMR program. This is one of three ways to adjust this value. The following priority is respected:

Note

The number of nodes is determined by the following (in order of priority):

- If `dmr_set_nodes_next_expand()` was called, its value is used.
- Else, if the `DMR_NODES_IN_EXPAND` environment variable is set, its value is used.
- Otherwise, the compile-time constant `NODES_IN_EXPAND` is used.

2.1.2.14 DMR_NODES_IN_SHRINK

```
#define DMR_NODES_IN_SHRINK 1
```

Default number of nodes to shrink by, if this functionality is available.

Sets how many nodes to remove each time a shrink operation is requested by default. This will only work if node removal is supported on the system and DMR is compiled with `JOBS_CAN_SHRINK` set to 1. If the default value exceeds the current number of nodes, it will be adjusted so that the resulting configuration has at least one node.

Note

The number of nodes to shrink by is determined by the following (in order of priority):

- If `dmr_set_nodes_next_shrink()` was called, its value is used.
- Else, if the `DMR_NODES_IN_SHRINK` environment variable is set, its value is used.
- Otherwise, the compile-time constant `DMR_NODES_IN_SHRINK` is used.

2.1.2.15 DMR_PRINT_ANALYTICS

```
#define DMR_PRINT_ANALYTICS 0
```

Whether or not to print analytics information.

Whether or not to print analytics information at runtime containing the following at each initialization: nodes, processes in current configuration, time since last reconfiguration

Note

You can set an environment variable of the same name when running to override the compiled value

2.1.2.16 DMR_PROCS_PER_NODE

```
#define DMR_PROCS_PER_NODE 1
```

Default number of processes per node in expander jobs.

Defines how many processes to spawn for each node in expander jobs. This is one of three ways to adjust this value. The following priority is respected:

Note

The number of processes per node is determined by the following (in order of priority):

- If `dmr_set_ppn_next_expand()` was called, its value is used.
- Else, if the `DMR_PROCS_PER_NODE` environment variable is set, its value is used.
- Otherwise, the compile-time constant `DMR_PROCS_PER_NODE` is used.

2.1.2.17 DMR_SKIP_SSH_CHECK

```
#define DMR_SKIP_SSH_CHECK 0
```

Whether or not to skip expander job ssh health check.

By default, expander jobs will perform a health check on new nodes to see if they are ready be connected to with ssh.

This setting is ignored with Slurm4DMR, as ssh health checks are never performed (resources are pre-allocated).

2.1.2.18 DMR_TALP_SENSITIVITY

```
#define DMR_TALP_SENSITIVITY 15
```

Sensitivity factor of DMR's performance-aware policy.

A higher value causes more resources to be added/removed in response to differences in communication efficiency w.r.t target efficiency

2.1.2.19 DMR_TALP_TARGET_CE

```
#define DMR_TALP_TARGET_CE 0.8
```

The target communication efficiency to aim for in the performance-aware policy.

Used when DMR's performance-aware policy is active. If the real value is found to be greater, the world will expand. If less, the world will shrink.

2.1.3 Typedef Documentation

2.1.3.1 DMRAction

```
typedef enum DMRActionEnum DMRAction
```

Used by DMR to tell user which action to take on current process.

Returned by the main dmr functions, indicating what step(s) the library user should take next to ensure correct functionality of the library.

2.1.3.2 DMRSuggestion

```
typedef enum DMRSuggestionEnum DMRSuggestion
```

Used by library user to suggest to DMR which action to take.

Tell DMR how to reconfigure, or to just stay in current configuration. The suggestion will immediately be acted on if DMR is not currently performing some other action.

2.1.4 Enumeration Type Documentation

2.1.4.1 DMRActionEnum

```
enum DMRActionEnum
```

Used by DMR to tell user which action to take on current process.

Returned by the main dmr functions, indicating what step(s) the library user should take next to ensure correct functionality of the library.

Enumerator

DMR_NO_ACTION	No action is required from the library user
DMR_RECONF	Should call dmr_reconfigure()
DMR_RESTART_RECONF	Should call restart logic to retrieve data, then call dmr_reconfigure()
DMR_REDIST_FINALIZE	Should call checkpoint or data redistribution logic to save data, then call dmr_finalize()
DMR_FINALIZE	Should call dmr_finalize()
DMR_CLEANUP	Optionally call internal cleanup logic
DMR_ERROR	Indicates an error has occurred

2.1.4.2 DMRSuggestionEnum

```
enum DMRSuggestionEnum
```

Used by library user to suggest to DMR which action to take.

Tell DMR how to reconfigure, or to just stay in current configuration. The suggestion will immediately be acted on if DMR is not currently performing some other action.

Enumerator

ROUND_POLICY	Double node count up until <code>dmr_get_policy_max_nodes()</code> , then minimize
CE_POLICY	Aim for a communication efficiency of DMR_TALP_TARGET_CE
SLURM4DMR_ROUND_POLICY	Expand up until <code>dmr_get_policy_max_nodes()</code> , then minimize
SLURM4DMR_CE_POLICY	Aim for a communication efficiency of DMR_TALP_TARGET_CE, considering cluster status
SLURM4DMR_QUEUE_POLICY	Try to stay at node preference but accept min and max nodes
SHOULD_EXPAND	Expand the world by <code>dmr_get_nodes_next_expand()</code> nodes
SHOULD_SHRINK	Shrink by <code>dmr_get_nodes_next_shrink()</code> nodes
SHOULD_STAY	Do nothing

2.1.5 Function Documentation

2.1.5.1 `dmr_cancel_expansion()`

```
void dmr_cancel_expansion ()
```

Cancel a pending expansion job.

Cancel an expansion job that has been requested. This function MUST be called by all the processes of the current DMR reconfiguration.

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with `dmr_init()`

2.1.5.2 `dmr_check()`

```
DMRAction dmr_check (
    DMR_Suggestion suggested_reconfiguration)
```

Suggest or check on DMR reconfiguration progress.

Call this function periodically to suggest a reconfiguration or check on reconfiguration progress. Takes a `DMR_Suggestion` value, which determines how DMR will try to reconfigure the world.

Parameters

in	<i>suggested_reconfiguration</i>	A suggestion to DMR as to how to reconfigure processes. See <code>DMR_Suggestion</code> definition.
----	----------------------------------	---

Precondition

All processes in MPI_COMM_WORLD call the function, providing the same [DMRSuggestion](#)
DMR has already been initialized with [dmr_init\(\)](#)
DMR has not already been finalized with [dmr_finalize\(\)](#)
DMR is not expecting some other action, as indicated by a returned [DMRAction](#)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for `dmr_check` are:

- `DMR_NO_ACTION`: returned when no further action is possible or desired at this time
- `DMR_RECONFIG`: returned when the environment is ready to be expanded or shrunk
- `DMR_ERROR`: returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.1.5.3 `dmr_finalize()`

```
DMRAction dmr_finalize ()
```

Finalize the DMR environment.

Call this function only when it is suggested by [dmr_check\(\)](#), or when finishing execution of the whole program. Clean up resources and, if necessary, kill any pending expander job that is no longer needed. Exit process if we are reconfiguring.

Warning

Process termination point when called in the context of a reconfiguration

Precondition

DMR has already been initialized with [dmr_init\(\)](#)
DMR has not already been finalized with [dmr_finalize\(\)](#)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for `dmr_finalize` are:

- `DMR_CLEANUP`: returned when called outside of a reconfiguration context
- `DMR_ERROR`: returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.1.5.4 dmr_get_active_expansions()

```
int dmr_get_active_expansions ()
```

Get the number of expansion jobs that are currently active.

Get the number of expansion jobs currently active in DMR. The count starts at 0, increments at every expansion, and decrements at every shrink. This function is safe to call at any point of execution, even before [dmr_init\(\)](#) has been called.

Returns

An integer representing the number of active expansion jobs, or -1 in case of failure

2.1.5.5 dmr_get_current_node_count()

```
int dmr_get_current_node_count ()
```

Get the number of nodes that are participating in the current DMR session.

Get a count of the nodes that are part of the current configuration of DMR

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.6 dmr_get_last_action()

```
DMRAction dmr_get_last_action (  
    void )
```

Get the last action that was returned by DMR.

Get the last [DMRAction](#) returned by any of the main DMR functions, e.g. [dmr_init](#), [dmr_check](#), [dmr_reconfigure](#), or [dmr_finalize](#).

Note

If no action has been returned yet, returns [DMR_NO_ACTION](#)

2.1.5.7 dmr_get_nodes_next_expand()

```
int dmr_get_nodes_next_expand ()
```

Get the number of nodes to request next time expanding.

Get the number of nodes to spawn onto the next time expanding. Only valid to call from the root process.

Precondition

The calling process is rank 0 in [MPI_COMM_WORLD](#)

DMR has already been initialized with [dmr_init\(\)](#)

Returns

The number of nodes to request next time expanding, or -1 in error cases

2.1.5.8 dmr_get_nodes_next_shrink()

```
int dmr_get_nodes_next_shrink ()
```

Get the number of nodes that would be removed if shrinking.

Get the number of nodes that would be removed from the program if a shrink was completed now.

Precondition

The calling process is rank 0 in MPI_COMM_WORLD

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.9 dmr_get_procs_next_expand()

```
int dmr_get_procs_next_expand ()
```

Get the number of processes to spawn next time expanding.

Get the number of processes to spawn the next time expanding. Only valid to call from the root process.

Precondition

The calling process is rank 0 in MPI_COMM_WORLD

DMR has already been initialized with [dmr_init\(\)](#)

Returns

The number of processes to spawn next time expanding, or -1 in error cases

2.1.5.10 dmr_get_procs_next_shrink()

```
int dmr_get_procs_next_shrink ()
```

Get the number of processes that would be removed if shrinking.

Get the number of processes that would be removed from the program if a shrink was requested now.

Precondition

The calling process is rank 0 in MPI_COMM_WORLD

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.11 dmr_get_reconfig_count()

```
int dmr_get_reconfig_count ()
```

Check how many times DMR has reconfigured itself since original launch.

Determines how many times since launch any resizing (shrinking or growing) has occurred, from the perspective of the calling process. This function is safe to call at any point of execution, even before [dmr_init\(\)](#) has been called.

Precondition

The environment variable DMR_RECONFIG_COUNT is set by the dmr_wrapper logic

Returns

0 if no expansions, a positive integer indicating the number of reconfigurations otherwise, or -1 in error cases

2.1.5.12 dmr_init()

```
DMRAction dmr_init (
    int argc,
    char * argv[])
```

Initialize the DMR library, unlocking other functionality.

Program starting point. Contains validation checks and initializers ensuring that the library will work as intended. Take argc and argv exactly as they exist in the main function of any C program.

Parameters

in	<i>argc</i>	Number of arguments in the argv array
in	<i>argv</i>	Array of arguments to pass on to any child processes (DMR assumes argv[0] is the executable)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for dmr_init are:

- [DMR_NO_ACTION](#): returned when DMR is initialized for the first time
- [DMR_RESTART_RECONFIG](#): returned when DMR is initialized after having expanded or shrunk the communicator
- [DMR_ERROR](#): returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.1.5.13 dmr_intercomm_available()

```
int dmr_intercomm_available ()
```

Check whether DMR is currently exposing a usable intercommunicator.

Check whether or not DMR_INTERCOMM is currently MPI_COMM_NULL or not. In cases where we are expanding or shrinking the world and we are compiled with CHECKPOINT_RESTART=0, the intercomm is available.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

Returns

1 if DMR_INTERCOMM is usable, or 0 otherwise

2.1.5.14 dmr_pending_expansion()

```
int dmr_pending_expansion ()
```

Indicate whether an expansion job is currently pending.

Get an indication of whether or not an expansion job is pending. This can be used to cancel the job safely using [dmr_cancel_expansion\(\)](#). The return value will not change unless dmr reconfiguration logic is called

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

Returns

1 if an expansion is pending, or 0 otherwise

2.1.5.15 dmr_reconfigure()

```
DMRAction dmr_reconfigure ()
```

Reconfigure the DMR environment.

Call this function only when it is suggested by [dmr_initialize\(\)](#) or [dmr_check\(\)](#). Reconfigure the MPI processes as necessary for the action in progress.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

DMR has not already been finalized with [dmr_finalize\(\)](#)

DMR is expecting a call to reconfigure, as indicated by a returned [DMRAction](#)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for [dmr_reconfigure](#) are:

- [DMR_NO_ACTION](#): returned by reconfigured processes when no further action is needed
- [DMR_REDIST_FINALIZE](#): returned on the communicator that is due to terminate when completing reconfiguration
- [DMR_ERROR](#): returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.1.5.16 dmr_set_jobs_next_shrink()

```
void dmr_set_jobs_next_shrink (
    int jobs)
```

Set the number of jobs to remove next time shrinking.

Set the count of whole jobs to remove when shrinking next time. Calls from non-zero processes are ignored.

Parameters

in	<i>procs</i>	Desired jobs to shrink by, which must be greater than 0 and less than or equal to dmr_get_active_expansions
----	--------------	---

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.17 dmr_set_nodes_next_expand()

```
void dmr_set_nodes_next_expand (
    int nodes)
```

Set the number of nodes to request next time expanding.

Set the number of nodes to launch onto the next time expanding. Calls from non-zero processes have no effect. This value overrides any value read from the compilation or environment, but resets after a reconfiguration has occurred. See [DMR_NODES_IN_EXPAND](#) for an explanation of the priority used.

Parameters

in	<i>nodes</i>	Number of nodes to request
----	--------------	----------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes

DMR is not already in the process of expanding

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.18 dmr_set_nodes_next_shrink()

```
void dmr_set_nodes_next_shrink (
    int nodes)
```

Set the number of nodes to remove next time shrinking.

Set the number of nodes to remove next time shrinking. Calls from non-zero processes are ignored. This feature is only available if we have the ability to shrink Slurm jobs. Calling this function overrides any value read from the compilation or the environment. See [NODES_IN_SHRINK](#) for an explanation of priority rules.

Parameters

in	<i>procs</i>	Desired nodes to shrink by
----	--------------	----------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes

DMR is compiled with JOBS_CAN_SHRINK == 1

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.19 dmr_set_policy_max_nodes()

```
void dmr_set_policy_max_nodes (  
    int nodes)
```

Set the maximum nodes when adding nodes by policy.

Set the maximum number of nodes to scale to when auto-sizing due to policy. Only applicable when a policy is in use which automatically adjust DMR program size.

Parameters

in	<i>nodes</i>	Maximum number of nodes
----	--------------	-------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.20 dmr_set_policy_min_nodes()

```
void dmr_set_policy_min_nodes (  
    int nodes)
```

Set the minimum nodes when removing nodes by policy.

Set the minimum number of nodes to scale to when auto-sizing due to policy. Only applicable when a policy is in use which automatically adjust DMR program size.

Parameters

in	<i>nodes</i>	Minimum number of nodes
----	--------------	-------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.21 dmr_set_policy_pref_nodes()

```
void dmr_set_policy_pref_nodes (
    int nodes)
```

Set the preferred number of nodes for applicable policies to use.

Set the preferred number of nodes the current program should be in. This will only have an effect for specific policies which can adjust to match a preference

Parameters

in	<i>nodes</i>	Preferred number of nodes
----	--------------	---------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.22 dmr_set_policy_stride()

```
void dmr_set_policy_stride (
    int multiplier)
```

Set the stride for applicable policies to use.

Set the multiplier for policy-based decisions to use. For example, if the multiplier is 2, expansions will double the size of the world and shrinks will halve it.

Parameters

in	<i>multiplier</i>	Value of the multiplier
----	-------------------	-------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.23 dmr_set_ppn_next_expand()

```
void dmr_set_ppn_next_expand (
    int ppn)
```

Set the number of processes per node to spawn next time expanding.

Set the number of processes to spawn the next time expanding. Calls from non-zero processes are ignored. This number will be multiplied by the number of nodes in the next expand. Setting this will override any value read from the environment or compilation default until the next reconfiguration. See [DMR_PROCS_PER_NODE](#) for an explanation of priority rules.

Parameters

in	<i>procs</i>	Desired process per node count
----	--------------	--------------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes
 DMR is not already in the process of expanding
 DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.24 dmr_set_procs_next_expand()

```
void dmr_set_procs_next_expand (
    int procs)
```

Set the EXACT number of processes to spawn next time expanding.

Set the total number of processes to spawn the next time expanding. Calls from non-zero processes are ignored. The total number set is divided between all the nodes in the next expand. Setting this overrides any value set in `dmr_set_ppn_next_expand`.

Parameters

in	<i>procs</i>	The number of processes to add
----	--------------	--------------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes
 DMR is not already in the process of expanding
 DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.25 dmr_set_procs_next_shrink()

```
void dmr_set_procs_next_shrink (
    int procs)
```

Set the number of processes to remove next time shrinking.

Set the number of processes to remove next time shrinking. This feature is only available if we have the ability to shrink Slurm jobs. Changing the process count to remove will affect the node count which we are removing in accordance with the new process count.

Parameters

in	<i>procs</i>	Desired processes to shrink by
----	--------------	--------------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes
 DMR is compiled with JOBS_CAN_SHRINK == 1
 DMR has already been initialized with [dmr_init\(\)](#)

2.1.5.26 dmr_set_reconf_step_inhibitor()

```
void dmr_set_reconf_step_inhibitor (
    int steps)
```

Set an inhibitor for requesting reconfigurations.

Set an inhibitor that will prevent DMR from requesting any new reconfiguration when `dmr_check` is called. If the inhibitor is `N`, then `N` out of every `N+1` calls to `dmr_check(...)` are ignored. The inhibitor has no effect if a reconfiguration is already underway.

Parameters

in	steps	Value of the inhibitor
----	-------	------------------------

Warning

All processes in `MPI_COMM_WORLD` must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.1.6 Variable Documentation

2.1.6.1 DMR_INTERCOMM

```
MPI_Comm DMR_INTERCOMM [extern]
```

The intercommunicator used for data reconfiguration.

The intercommunicator connecting the processes of each DMR reconfiguration, enabling redistribution of data over MPI functionality.

Not to be used unless suggested to by a [DMRAction](#) (`DMR_RESTART_RECONF` or `REDIST_FINALIZE`). A helper function, [dmr_intercomm_available\(\)](#), is provided.

2.2 src/dmr.c File Reference

DMR main functionality.

```
#include "dmr.h"
#include "dmr_internal.h"
```

Functions

- [DMRAction dmr_init](#) (int argc, char *argv[])
Initialize the DMR library, unlocking other functionality.
- [DMRAction dmr_check](#) ([DMRSuggestion](#) suggested_reconfiguration)
Suggest or check on DMR reconfiguration progress.
- [DMRAction dmr_reconfigure](#) (void)
Reconfigure the DMR environment.
- [DMRAction dmr_finalize](#) (void)
Finalize the DMR environment.
- int [dmr_intercomm_available](#) (void)
Check whether DMR is currently exposing a usable intercommunicator.
- int [dmr_get_reconfig_count](#) (void)
Check how many times DMR has reconfigured itself since original launch.
- int [dmr_get_active_expansions](#) (void)
Get the number of expansion jobs that are currently active.
- int [dmr_get_nodes_next_expand](#) (void)
Get the number of nodes to request next time expanding.
- void [dmr_set_nodes_next_expand](#) (int nodes)
Set the number of nodes to request next time expanding.
- int [dmr_get_procs_next_expand](#) (void)
Get the number of processes to spawn next time expanding.
- void [dmr_set_procs_next_expand](#) (int procs)
Set the EXACT number of processes to spawn next time expanding.
- void [dmr_set_ppn_next_expand](#) (int ppn)
Set the number of processes per node to spawn next time expanding.
- int [dmr_get_procs_next_shrink](#) (void)
Get the number of processes that would be removed if shrinking.
- void [dmr_set_procs_next_shrink](#) (int procs)
Set the number of processes to remove next time shrinking.
- int [dmr_get_nodes_next_shrink](#) (void)
Get the number of nodes that would be removed if shrinking.
- void [dmr_set_nodes_next_shrink](#) (int nodes)
Set the number of nodes to remove next time shrinking.
- void [dmr_set_jobs_next_shrink](#) (int jobs)
Set the number of jobs to remove next time shrinking.
- int [dmr_get_current_node_count](#) (void)
Get the number of nodes that are participating in the current DMR session.
- int [dmr_pending_expansion](#) (void)
Indicate whether an expansion job is currently pending.
- void [dmr_cancel_expansion](#) (void)
Cancel a pending expansion job.
- void [dmr_set_reconf_step_inhibitor](#) (int steps)
Set an inhibitor for requesting reconfigurations.
- void [dmr_set_policy_min_nodes](#) (int nodes)
Set the minimum nodes when removing nodes by policy.
- void [dmr_set_policy_max_nodes](#) (int nodes)
Set the maximum nodes when adding nodes by policy.
- void [dmr_set_policy_stride](#) (int multiplier)
Set the stride for applicable policies to use.
- void [dmr_set_policy_pref_nodes](#) (int nodes)

Set the preferred number of nodes for applicable policies to use.

- `DMRAction dmr_get_last_action` (void)
Get the last action that was returned by DMR.
- `DMRState * dmr_get_state` (void)
Get the current DMRState, for testing.
- `void dmr_set_state` (DMRState *state)
Set the current DMRState, for testing.
- `DMRControllerState * dmr_get_controller_state` (void)
Get the controller state of the DMR application.
- `void dmr_set_controller_state` (DMRControllerState *custom_controller)
Override the controller state with a testing state.

Variables

- `MPI_Comm DMR_INTERCOMM` = `MPI_COMM_NULL`
The intercommunicator used for data reconfiguration.

2.2.1 Detailed Description

DMR main functionality.

Contains the publicly accessible interface of the DMR library

2.2.2 Function Documentation

2.2.2.1 dmr_cancel_expansion()

```
void dmr_cancel_expansion ()
```

Cancel a pending expansion job.

Cancel an expansion job that has been requested. This function MUST be called by all the processes of the current DMR reconfiguration.

Warning

All processes in `MPI_COMM_WORLD` must call the function.

Precondition

DMR has already been initialized with `dmr_init()`

2.2.2.2 dmr_check()

```
DMRAction dmr_check (
    DMRuggestion suggested_reconfiguration)
```

Suggest or check on DMR reconfiguration progress.

Call this function periodically to suggest a reconfiguration or check on reconfiguration progress. Takes a `DMRuggestion` value, which determines how DMR will try to reconfigure the world.

Parameters

in	<i>suggested_reconfiguration</i>	A suggestion to DMR as to how to reconfigure processes. See DMRSuggestion definition.
----	----------------------------------	---

Precondition

All processes in MPI_COMM_WORLD call the function, proviing the same [DMRSuggestion](#)
 DMR has already been initialized with [dmr_init\(\)](#)
 DMR has not already been finalized with [dmr_finalize\(\)](#)
 DMR is not expecting some other action, as indicated by a returned [DMRAction](#)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for `dmr_check` are:

- `DMR_NO_ACTION`: returned when no further action is possible or desired at this time
- `DMR_RECONFIG`: returned when the environment is ready to be expanded or shrunk
- `DMR_ERROR`: returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.2.2.3 dmr_finalize()

```
DMRAction dmr_finalize ()
```

Finalize the DMR environment.

Call this function only when it is suggested by [dmr_check\(\)](#), or when finishing execution of the whole program. Clean up resources and, if necessary, kill any pending expander job that is no longer needed. Exit process if we are reconfiguring.

Warning

Process termination point when called in the context of a reconfiguration

Precondition

DMR has already been initialized with [dmr_init\(\)](#)
 DMR has not already been finalized with [dmr_finalize\(\)](#)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for `dmr_finalize` are:

- `DMR_CLEANUP`: returned when called outside of a reconfiguration context
- `DMR_ERROR`: returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.2.2.4 dmr_get_active_expansions()

```
int dmr_get_active_expansions ()
```

Get the number of expansion jobs that are currently active.

Get the number of expansion jobs currently active in DMR. The count starts at 0, increments at every expansion, and decrements at every shrink. This function is safe to call at any point of execution, even before [dmr_init\(\)](#) has been called.

Returns

An integer representing the number of active expansion jobs, or -1 in case of failure

2.2.2.5 dmr_get_current_node_count()

```
int dmr_get_current_node_count ()
```

Get the number of nodes that are participating in the current DMR session.

Get a count of the nodes that are part of the current configuration of DMR

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.6 dmr_get_last_action()

```
DMRAction dmr_get_last_action (  
    void )
```

Get the last action that was returned by DMR.

Get the last [DMRAction](#) returned by any of the main DMR functions, e.g. [dmr_init](#), [dmr_check](#), [dmr_reconfigure](#), or [dmr_finalize](#).

Note

If no action has been returned yet, returns [DMR_NO_ACTION](#)

2.2.2.7 dmr_get_nodes_next_expand()

```
int dmr_get_nodes_next_expand ()
```

Get the number of nodes to request next time expanding.

Get the number of nodes to spawn onto the next time expanding. Only valid to call from the root process.

Precondition

The calling process is rank 0 in [MPI_COMM_WORLD](#)

DMR has already been initialized with [dmr_init\(\)](#)

Returns

The number of nodes to request next time expanding, or -1 in error cases

2.2.2.8 dmr_get_nodes_next_shrink()

```
int dmr_get_nodes_next_shrink ()
```

Get the number of nodes that would be removed if shrinking.

Get the number of nodes that would be removed from the program if a shrink was completed now.

Precondition

The calling process is rank 0 in MPI_COMM_WORLD

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.9 dmr_get_procs_next_expand()

```
int dmr_get_procs_next_expand ()
```

Get the number of processes to spawn next time expanding.

Get the number of processes to spawn the next time expanding. Only valid to call from the root process.

Precondition

The calling process is rank 0 in MPI_COMM_WORLD

DMR has already been initialized with [dmr_init\(\)](#)

Returns

The number of processes to spawn next time expanding, or -1 in error cases

2.2.2.10 dmr_get_procs_next_shrink()

```
int dmr_get_procs_next_shrink ()
```

Get the number of processes that would be removed if shrinking.

Get the number of processes that would be removed from the program if a shrink was requested now.

Precondition

The calling process is rank 0 in MPI_COMM_WORLD

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.11 dmr_get_reconfig_count()

```
int dmr_get_reconfig_count ()
```

Check how many times DMR has reconfigured itself since original launch.

Determines how many times since launch any resizing (shrinking or growing) has occurred, from the perspective of the calling process. This function is safe to call at any point of execution, even before [dmr_init\(\)](#) has been called.

Precondition

The environment variable DMR_RECONFIG_COUNT is set by the dmr_wrapper logic

Returns

0 if no expansions, a positive integer indicating the number of reconfigurations otherwise, or -1 in error cases

2.2.2.12 dmr_init()

```
DMRAction dmr_init (
    int argc,
    char * argv[])
```

Initialize the DMR library, unlocking other functionality.

Program starting point. Contains validation checks and initializers ensuring that the library will work as intended. Take argc and argv exactly as they exist in the main function of any C program.

Parameters

in	<i>argc</i>	Number of arguments in the argv array
in	<i>argv</i>	Array of arguments to pass on to any child processes (DMR assumes argv[0] is the executable)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for dmr_init are:

- [DMR_NO_ACTION](#): returned when DMR is initialized for the first time
- [DMR_RESTART_RECONFIG](#): returned when DMR is initialized after having expanded or shrunk the communicator
- [DMR_ERROR](#): returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.2.2.13 dmr_intercomm_available()

```
int dmr_intercomm_available ()
```

Check whether DMR is currently exposing a usable intercommunicator.

Check whether or not DMR_INTERCOMM is currently MPI_COMM_NULL or not. In cases where we are expanding or shrinking the world and we are compiled with CHECKPOINT_RESTART=0, the intercomm is available.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

Returns

1 if DMR_INTERCOMM is usable, or 0 otherwise

2.2.2.14 dmr_pending_expansion()

```
int dmr_pending_expansion ()
```

Indicate whether an expansion job is currently pending.

Get an indication of whether or not an expansion job is pending. This can be used to cancel the job safely using [dmr_cancel_expansion\(\)](#). The return value will not change unless dmr reconfiguration logic is called

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

Returns

1 if an expansion is pending, or 0 otherwise

2.2.2.15 dmr_reconfigure()

```
DMRAction dmr_reconfigure ()
```

Reconfigure the DMR environment.

Call this function only when it is suggested by [dmr_initialize\(\)](#) or [dmr_check\(\)](#). Reconfigure the MPI processes as necessary for the action in progress.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

DMR has not already been finalized with [dmr_finalize\(\)](#)

DMR is expecting a call to reconfigure, as indicated by a returned [DMRAction](#)

Returns

A [DMRAction](#) enum value for the user to act on. Possible return values for [dmr_reconfigure](#) are:

- [DMR_NO_ACTION](#): returned by reconfigured processes when no further action is needed
- [DMR_REDIST_FINALIZE](#): returned on the communicator that is due to terminate when completing reconfiguration
- [DMR_ERROR](#): returned in error cases

See the [DMRAction](#) definition for an explanation of the meaning of each action.

2.2.2.16 dmr_set_jobs_next_shrink()

```
void dmr_set_jobs_next_shrink (
    int jobs)
```

Set the number of jobs to remove next time shrinking.

Set the count of whole jobs to remove when shrinking next time. Calls from non-zero processes are ignored.

Parameters

in	<i>procs</i>	Desired jobs to shrink by, which must be greater than 0 and less than or equal to dmr_get_active_expansions
----	--------------	---

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.17 dmr_set_nodes_next_expand()

```
void dmr_set_nodes_next_expand (
    int nodes)
```

Set the number of nodes to request next time expanding.

Set the number of nodes to launch onto the next time expanding. Calls from non-zero processes have no effect. This value overrides any value read from the compilation or environment, but resets after a reconfiguration has occurred. See [DMR_NODES_IN_EXPAND](#) for an explanation of the priority used.

Parameters

in	<i>nodes</i>	Number of nodes to request
----	--------------	----------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes

DMR is not already in the process of expanding

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.18 dmr_set_nodes_next_shrink()

```
void dmr_set_nodes_next_shrink (
    int nodes)
```

Set the number of nodes to remove next time shrinking.

Set the number of nodes to remove next time shrinking. Calls from non-zero processes are ignored. This feature is only available if we have the ability to shrink Slurm jobs. Calling this function overrides any value read from the compilation or the environment. See [NODES_IN_SHRINK](#) for an explanation of priority rules.

Parameters

in	<i>procs</i>	Desired nodes to shrink by
----	--------------	----------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes

DMR is compiled with JOBS_CAN_SHRINK == 1

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.19 dmr_set_policy_max_nodes()

```
void dmr_set_policy_max_nodes (
    int nodes)
```

Set the maximum nodes when adding nodes by policy.

Set the maximum number of nodes to scale to when auto-sizing due to policy. Only applicable when a policy is in use which automatically adjust DMR program size.

Parameters

in	<i>nodes</i>	Maximum number of nodes
----	--------------	-------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.20 dmr_set_policy_min_nodes()

```
void dmr_set_policy_min_nodes (
    int nodes)
```

Set the minimum nodes when removing nodes by policy.

Set the minimum number of nodes to scale to when auto-sizing due to policy. Only applicable when a policy is in use which automatically adjust DMR program size.

Parameters

in	<i>nodes</i>	Minimum number of nodes
----	--------------	-------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.21 dmr_set_policy_pref_nodes()

```
void dmr_set_policy_pref_nodes (
    int nodes)
```

Set the preferred number of nodes for applicable policies to use.

Set the preferred number of nodes the current program should be in. This will only have an effect for specific policies which can adjust to match a preference

Parameters

in	<i>nodes</i>	Preferred number of nodes
----	--------------	---------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.22 dmr_set_policy_stride()

```
void dmr_set_policy_stride (
    int multiplier)
```

Set the stride for applicable policies to use.

Set the multiplier for policy-based decisions to use. For example, if the multiplier is 2, expansions will double the size of the world and shrinks will halve it.

Parameters

in	<i>multiplier</i>	Value of the multiplier
----	-------------------	-------------------------

Warning

All processes in MPI_COMM_WORLD must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.23 dmr_set_ppn_next_expand()

```
void dmr_set_ppn_next_expand (
    int ppn)
```

Set the number of processes per node to spawn next time expanding.

Set the number of processes to spawn the next time expanding. Calls from non-zero processes are ignored. This number will be multiplied by the number of nodes in the next expand. Setting this will override any value read from the environment or compilation default until the next reconfiguration. See [DMR_PROCS_PER_NODE](#) for an explanation of priority rules.

Parameters

in	<i>procs</i>	Desired process per node count
----	--------------	--------------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes
 DMR is not already in the process of expanding
 DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.24 dmr_set_procs_next_expand()

```
void dmr_set_procs_next_expand (
    int procs)
```

Set the EXACT number of processes to spawn next time expanding.

Set the total number of processes to spawn the next time expanding. Calls from non-zero processes are ignored. The total number set is divided between all the nodes in the next expand. Setting this overrides any value set in `dmr_set_ppn_next_expand`.

Parameters

in	<i>procs</i>	The number of processes to add
----	--------------	--------------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes
 DMR is not already in the process of expanding
 DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.25 dmr_set_procs_next_shrink()

```
void dmr_set_procs_next_shrink (
    int procs)
```

Set the number of processes to remove next time shrinking.

Set the number of processes to remove next time shrinking. This feature is only available if we have the ability to shrink Slurm jobs. Changing the process count to remove will affect the node count which we are removing in accordance with the new process count.

Parameters

in	<i>procs</i>	Desired processes to shrink by
----	--------------	--------------------------------

Precondition

Rank 0 in MPI_COMM_WORLD is among the calling processes
 DMR is compiled with JOBS_CAN_SHRINK == 1
 DMR has already been initialized with [dmr_init\(\)](#)

2.2.2.26 dmr_set_reconf_step_inhibitor()

```
void dmr_set_reconf_step_inhibitor (
    int steps)
```

Set an inhibitor for requesting reconfigurations.

Set an inhibitor that will prevent DMR from requesting any new reconfiguration when `dmr_check` is called. If the inhibitor is `N`, then `N` out of every `N+1` calls to `dmr_check(...)` are ignored. The inhibitor has no effect if a reconfiguration is already underway.

Parameters

in	steps	Value of the inhibitor
----	-------	------------------------

Warning

All processes in `MPI_COMM_WORLD` must call the function.

Precondition

DMR has already been initialized with [dmr_init\(\)](#)

2.2.3 Variable Documentation

2.2.3.1 DMR_INTERCOMM

```
MPI_Comm DMR_INTERCOMM = MPI_COMM_NULL
```

The intercommunicator used for data reconfiguration.

The intercommunicator connecting the processes of each DMR reconfiguration, enabling redistribution of data over MPI functionality.

Not to be used unless suggested to by a [DMRAction](#) (`DMR_RESTART_RECONF` or `REDIST_FINALIZE`). A helper function, [dmr_intercomm_available\(\)](#), is provided.

Index

CE_POLICY

dmr.h, [12](#)

CUSTOM_SLURM_BIN_PREFIX

dmr.h, [6](#)

dmr.c

dmr_cancel_expansion, [24](#)

dmr_check, [24](#)

dmr_finalize, [25](#)

dmr_get_active_expansions, [25](#)

dmr_get_current_node_count, [26](#)

dmr_get_last_action, [26](#)

dmr_get_nodes_next_expand, [26](#)

dmr_get_nodes_next_shrink, [26](#)

dmr_get_procs_next_expand, [27](#)

dmr_get_procs_next_shrink, [27](#)

dmr_get_reconfig_count, [27](#)

dmr_init, [28](#)

DMR_INTERCOMM, [34](#)

dmr_intercomm_available, [28](#)

dmr_pending_expansion, [29](#)

dmr_reconfigure, [29](#)

dmr_set_jobs_next_shrink, [29](#)

dmr_set_nodes_next_expand, [30](#)

dmr_set_nodes_next_shrink, [30](#)

dmr_set_policy_max_nodes, [31](#)

dmr_set_policy_min_nodes, [31](#)

dmr_set_policy_pref_nodes, [31](#)

dmr_set_policy_stride, [32](#)

dmr_set_ppn_next_expand, [32](#)

dmr_set_procs_next_expand, [33](#)

dmr_set_procs_next_shrink, [33](#)

dmr_set_reconf_step_inhibitor, [33](#)

dmr.h

CE_POLICY, [12](#)

CUSTOM_SLURM_BIN_PREFIX, [6](#)

DMR_AUTO, [6](#)

DMR_BLOCKING_REQ, [6](#)

dmr_cancel_expansion, [12](#)

dmr_check, [12](#)

DMR_CHECKPOINT_RESTART, [6](#)

DMR_CLEANUP, [11](#)

DMR_DEBUG_LEVEL, [7](#)

DMR_DEFAULT_INHIBITOR, [7](#)

DMR_DEFAULT_POLICY_MAX, [7](#)

DMR_DEFAULT_POLICY_MIN, [7](#)

DMR_DEFAULT_POLICY_PREF, [8](#)

DMR_DEFAULT_POLICY_STRIDE, [8](#)

DMR_ERROR, [11](#)

DMR_FINALIZE, [11](#)

dmr_finalize, [13](#)

dmr_get_active_expansions, [13](#)

dmr_get_current_node_count, [14](#)

dmr_get_last_action, [14](#)

dmr_get_nodes_next_expand, [14](#)

dmr_get_nodes_next_shrink, [14](#)

dmr_get_procs_next_expand, [15](#)

dmr_get_procs_next_shrink, [15](#)

dmr_get_reconfig_count, [15](#)

dmr_init, [16](#)

DMR_INTERCOMM, [22](#)

dmr_intercomm_available, [16](#)

DMR_JOBS_CAN_GROW, [8](#)

DMR_JOBS_CAN_SHRINK, [8](#)

DMR_NO_ACTION, [11](#)

DMR_NODES_IN_EXPAND, [8](#)

DMR_NODES_IN_SHRINK, [9](#)

dmr_pending_expansion, [17](#)

DMR_PRINT_ANALYTICS, [9](#)

DMR_PROCS_PER_NODE, [9](#)

DMR_RECONF, [11](#)

dmr_reconfigure, [17](#)

DMR_REDIST_FINALIZE, [11](#)

DMR_RESTART_RECONF, [11](#)

dmr_set_jobs_next_shrink, [17](#)

dmr_set_nodes_next_expand, [18](#)

dmr_set_nodes_next_shrink, [18](#)

dmr_set_policy_max_nodes, [19](#)

dmr_set_policy_min_nodes, [19](#)

dmr_set_policy_pref_nodes, [19](#)

dmr_set_policy_stride, [20](#)

dmr_set_ppn_next_expand, [20](#)

dmr_set_procs_next_expand, [21](#)

dmr_set_procs_next_shrink, [21](#)

dmr_set_reconf_step_inhibitor, [21](#)

DMR_SKIP_SSH_CHECK, [10](#)

DMR_TALP_SENSITIVITY, [10](#)

DMR_TALP_TARGET_CE, [10](#)

DMRAction, [11](#)

DMRActionEnum, [11](#)

DMRSuggestion, [11](#)

DMRSuggestionEnum, [11](#)

ROUND_POLICY, [12](#)

SHOULD_EXPAND, [12](#)

SHOULD_SHRINK, [12](#)

SHOULD_STAY, [12](#)

SLURM4DMR_CE_POLICY, [12](#)

SLURM4DMR_QUEUE_POLICY, [12](#)

SLURM4DMR_ROUND_POLICY, [12](#)

DMR_AUTO
 dmr.h, 6
 DMR_BLOCKING_REQ
 dmr.h, 6
 dmr_cancel_expansion
 dmr.c, 24
 dmr.h, 12
 dmr_check
 dmr.c, 24
 dmr.h, 12
 DMR_CHECKPOINT_RESTART
 dmr.h, 6
 DMR_CLEANUP
 dmr.h, 11
 DMR_DEBUG_LEVEL
 dmr.h, 7
 DMR_DEFAULT_INHIBITOR
 dmr.h, 7
 DMR_DEFAULT_POLICY_MAX
 dmr.h, 7
 DMR_DEFAULT_POLICY_MIN
 dmr.h, 7
 DMR_DEFAULT_POLICY_PREF
 dmr.h, 8
 DMR_DEFAULT_POLICY_STRIDE
 dmr.h, 8
 DMR_ERROR
 dmr.h, 11
 DMR_FINALIZE
 dmr.h, 11
 dmr_finalize
 dmr.c, 25
 dmr.h, 13
 dmr_get_active_expansions
 dmr.c, 25
 dmr.h, 13
 dmr_get_current_node_count
 dmr.c, 26
 dmr.h, 14
 dmr_get_last_action
 dmr.c, 26
 dmr.h, 14
 dmr_get_nodes_next_expand
 dmr.c, 26
 dmr.h, 14
 dmr_get_nodes_next_shrink
 dmr.c, 26
 dmr.h, 14
 dmr_get_procs_next_expand
 dmr.c, 27
 dmr.h, 15
 dmr_get_procs_next_shrink
 dmr.c, 27
 dmr.h, 15
 dmr_get_reconfig_count
 dmr.c, 27
 dmr.h, 15
 dmr_init
 dmr.c, 28
 dmr.h, 16
 DMR_INTERCOMM
 dmr.c, 34
 dmr.h, 22
 dmr_intercomm_available
 dmr.c, 28
 dmr.h, 16
 DMR_JOBS_CAN_GROW
 dmr.h, 8
 DMR_JOBS_CAN_SHRINK
 dmr.h, 8
 DMR_NO_ACTION
 dmr.h, 11
 DMR_NODES_IN_EXPAND
 dmr.h, 8
 DMR_NODES_IN_SHRINK
 dmr.h, 9
 dmr_pending_expansion
 dmr.c, 29
 dmr.h, 17
 DMR_PRINT_ANALYTICS
 dmr.h, 9
 DMR_PROCS_PER_NODE
 dmr.h, 9
 DMR_RECONF
 dmr.h, 11
 dmr_reconfigure
 dmr.c, 29
 dmr.h, 17
 DMR_REDIST_FINALIZE
 dmr.h, 11
 DMR_RESTART_RECONF
 dmr.h, 11
 dmr_set_jobs_next_shrink
 dmr.c, 29
 dmr.h, 17
 dmr_set_nodes_next_expand
 dmr.c, 30
 dmr.h, 18
 dmr_set_nodes_next_shrink
 dmr.c, 30
 dmr.h, 18
 dmr_set_policy_max_nodes
 dmr.c, 31
 dmr.h, 19
 dmr_set_policy_min_nodes
 dmr.c, 31
 dmr.h, 19
 dmr_set_policy_pref_nodes
 dmr.c, 31
 dmr.h, 19
 dmr_set_policy_stride
 dmr.c, 32
 dmr.h, 20
 dmr_set_ppn_next_expand
 dmr.c, 32
 dmr.h, 20

- dmr_set_procs_next_expand
 - dmr.c, [33](#)
 - dmr.h, [21](#)
- dmr_set_procs_next_shrink
 - dmr.c, [33](#)
 - dmr.h, [21](#)
- dmr_set_reconf_step_inhibitor
 - dmr.c, [33](#)
 - dmr.h, [21](#)
- DMR_SKIP_SSH_CHECK
 - dmr.h, [10](#)
- DMR_TALP_SENSITIVITY
 - dmr.h, [10](#)
- DMR_TALP_TARGET_CE
 - dmr.h, [10](#)
- DMRAction
 - dmr.h, [11](#)
- DMRActionEnum
 - dmr.h, [11](#)
- DMRSuggestion
 - dmr.h, [11](#)
- DMRSuggestionEnum
 - dmr.h, [11](#)
- include/dmr.h, [3](#)
- ROUND_POLICY
 - dmr.h, [12](#)
- SHOULD_EXPAND
 - dmr.h, [12](#)
- SHOULD_SHRINK
 - dmr.h, [12](#)
- SHOULD_STAY
 - dmr.h, [12](#)
- SLURM4DMR_CE_POLICY
 - dmr.h, [12](#)
- SLURM4DMR_QUEUE_POLICY
 - dmr.h, [12](#)
- SLURM4DMR_ROUND_POLICY
 - dmr.h, [12](#)
- src/dmr.c, [22](#)