

**NAME**

slurm\_create\_partition, slurm\_create\_reservation, slurm\_delete\_partition, slurm\_delete\_reservation, slurm\_init\_part\_desc\_msg, slurm\_init\_resv\_desc\_msg, slurm\_reconfigure, slurm\_shutdown, slurm\_takeover, slurm\_init\_update\_node\_msg, slurm\_update\_node, slurm\_update\_partition, slurm\_update\_reservation – Slurm administrative functions

**SYNTAX**

```
#include <slurm/slurm.h>

int slurm_create_partition (
    update_part_msg_t *update_part_msg_ptr
);

int slurm_create_reservation (
    resv_desc_msg_t *update_resv_msg_ptr
);

int slurm_delete_partition (
    delete_part_msg_t *delete_part_msg_ptr
);

int slurm_delete_reservation (
    reservation_name_msg_t *delete_resv_msg_ptr
);

void slurm_init_front_end_msg (
    update_front_end_msg_t *update_front_end_msg_ptr
);

void slurm_init_part_desc_msg (
    update_part_msg_t *update_part_msg_ptr
);

void slurm_init_resv_desc_msg (
    resv_desc_msg_t *update_resv_msg_ptr
);

void slurm_init_update_node_msg(
    update_node_msg_t *update_node_msg_ptr
);

int slurm_reconfigure ( );

int slurm_shutdown (
    uint16_t shutdown_options
);

int slurm_takeover ( );

int slurm_update_front_end (
    update_front_end_msg_t *update_front_end_msg_ptr
);

int slurm_update_node (
    update_node_msg_t *update_node_msg_ptr
);

int slurm_update_partition (
    update_part_msg_t *update_part_msg_ptr
);

int slurm_update_reservation (
    resv_desc_msg_t *update_resv_msg_ptr
);
```

**ARGUMENTS**

*shutdown\_options*  
 0: all slurm daemons are shutdown  
 1: slurmcld generates a core file



2: only the slurmcld is shutdown (no core file)

*delete\_part\_msg\_ptr*

Specifies the pointer to a partition delete request specification. See slurm.h for full details on the data structure's contents.

*delete\_resv\_msg\_ptr*

Specifies the pointer to a reservation delete request specification. See slurm.h for full details on the data structure's contents.

*update\_front\_end\_msg\_ptr*

Specifies the pointer to a front end node update request specification. See slurm.h for full details on the data structure's contents.

*update\_node\_msg\_ptr*

Specifies the pointer to a node update request specification. See slurm.h for full details on the data structure's contents.

*update\_part\_msg\_ptr*

Specifies the pointer to a partition create or update request specification. See slurm.h for full details on the data structure's contents.

*update\_resv\_msg\_ptr*

Specifies the pointer to a reservation create or update request specification. See slurm.h for full details on the data structure's contents.

## DESCRIPTION

**slurm\_create\_partition** Request that a new partition be created. Initialize the data structure using the **slurm\_init\_part\_desc\_msg** function prior to setting values of the parameters to be changed. Note: **slurm\_init\_part\_desc\_msg** is not equivalent to setting the data structure values to zero. A partition name must be set for the call to succeed. This function may only be successfully executed by user root.

**slurm\_create\_reservation** Request that a new reservation be created. Initialize the data structure using the **slurm\_init\_resv\_desc\_msg** function prior to setting values of the parameters to be changed. Note: **slurm\_init\_resv\_desc\_msg** is not equivalent to setting the data structure values to zero. The reservation's time limits, user or account restrictions, and node names or a node count must be specified for the call to succeed. This function may only be successfully executed by user root.

**slurm\_delete\_partition** Request that the specified partition be deleted. All jobs associated with the identified partition will be terminated and purged. This function may only be successfully executed by user root.

**slurm\_delete\_reservation** Request that the specified reservation be deleted. This function may only be successfully executed by user root.

**slurm\_init\_update\_front\_end\_msg** Initialize the contents of an update front end node descriptor with default values. Note: **slurm\_init\_update\_front\_end\_msg** is not equivalent to setting the data structure values to zero. Execute this function before executing **slurm\_update\_front\_end**.

**slurm\_init\_part\_desc\_msg** Initialize the contents of a partition descriptor with default values. Note: **slurm\_init\_part\_desc\_msg** is not equivalent to setting the data structure values to zero. Execute this function before executing **slurm\_create\_partition** or **slurm\_update\_partition**.

**slurm\_init\_resv\_desc\_msg** Initialize the contents of a reservation descriptor with default values. Note: **slurm\_init\_resv\_desc\_msg** is not equivalent to setting the data structure values to zero. Execute this function before executing **slurm\_create\_reservation** or **slurm\_update\_reservation**.

**slurm\_init\_update\_node\_msg** Initialize the contents of an update node descriptor with default values. Note: **slurm\_init\_update\_node\_msg** is not equivalent to setting the data structure values to zero. Execute this function before executing **slurm\_update\_node**.

**slurm\_reconfigure** Request that the Slurm controller re-read its configuration file. The new configuration parameters take effect immediately. This function may only be successfully executed by user root.

**slurm\_shutdown** Request that the Slurm controller terminate. This function may only be successfully executed by user root.

**slurm\_takeover** Request that the Slurm primary controller shutdown immediately and the backup



controller take over. This function may only be successfully executed by user root.

**slurm\_update\_front\_end** Request that the state of one or more front end nodes be updated. This function may only be successfully executed by user root. If used by some autonomous program, the state value most likely to be used is **NODE\_STATE\_DRAIN**.

**slurm\_update\_node** Request that the state of one or more nodes be updated. Note that the state of a node (e.g. DRAINING, IDLE, etc.) may be changed, but its hardware configuration may not be changed by this function. If the hardware configuration of a node changes, update the Slurm configuration file and execute the **slurm\_reconfigure** function. This function may only be successfully executed by user root. If used by some autonomous program, the state value most likely to be used is **NODE\_STATE\_DRAIN** or **NODE\_STATE\_FAILING**. The node state flag **NODE\_STATE\_NO\_RESPOND** may be specified without changing the underlying node state. Note that the node's **NODE\_STATE\_NO\_RESPOND** flag will be cleared as soon as the slurmd daemon on that node communicates with the slurmctld daemon. Likewise the state **NODE\_STATE\_DOWN** indicates that the slurmd daemon is not responding (and has not responded for an interval at least as long as the **SlurmdTimeout** configuration parameter). The node will leave the **NODE\_STATE\_DOWN** state as soon as the slurmd daemon communicates.

**slurm\_update\_partition** Request that the configuration of a partition be updated. Note that most, but not all parameters of a partition may be changed by this function. Initialize the data structure using the **slurm\_init\_part\_desc\_msg** function prior to setting values of the parameters to be changed. Note: **slurm\_init\_part\_desc\_msg** is not equivalent to setting the data structure values to zero. This function may only be successfully executed by user root.

**slurm\_update\_reservation** Request that the configuration of a reservation be updated. Initialize the data structure using the **slurm\_init\_resv\_desc\_msg** function prior to setting values of the parameters to be changed. Note: **slurm\_init\_resv\_desc\_msg** is not equivalent to setting the data structure values to zero. This function may only be successfully executed by user root.

## RETURN VALUE

On success, zero is returned. On error, -1 is returned, and the Slurm error code is set appropriately.

Exception: A successful **slurm\_create\_reservation** call returns a string containing the name of the reservation, in memory to be freed by the caller. A failed call returns NULL and sets the Slurm error code.

## ERRORS

**SLURM\_PROTOCOL\_VERSION\_ERROR** Protocol version has changed, re-link your code.

**ESLURM\_INVALID\_NODE\_NAME** The requested node name(s) is/are not valid.

**ESLURM\_INVALID\_NODE\_STATE** The specified state node state or requested node state transition is not valid.

**ESLURM\_INVALID\_PARTITION\_NAME** The requested partition name is not valid.

**ESLURM\_INVALID\_AUTHTYPE\_CHANGE** The **AuthType** parameter can not be changed using the **slurm\_reconfigure** function, but all Slurm daemons and commands must be restarted. See **slurm.conf(5)** for more information.

**ESLURM\_INVALID\_SCHEDTYPE\_CHANGE** The **SchedulerType** parameter can not be changed using the **slurm\_reconfigure** function, but the **slurmctld** daemon must be restarted. Manual changes to existing job parameters may also be required. See **slurm.conf(5)** for more information.

**ESLURM\_INVALID\_SWITCHTYPE\_CHANGE** The **SwitchType** parameter can not be changed using the **slurm\_reconfigure** function, but all Slurm daemons and commands must be restarted. All previously running jobs will be lost. See **slurm.conf(5)** for more information.

**ESLURM\_ACCESS\_DENIED** The requesting user lacks authorization for the requested action (e.g. trying to delete or modify another user's job).

**SLURM\_PROTOCOL\_SOCKET\_IMPL\_TIMEOUT** Timeout in communicating with Slurm controller.

**ESLURM\_RESERVATION\_ACCESS** Requestor is not authorized to access the reservation.

**ESLURM\_RESERVATION\_INVALID** Invalid reservation parameter given, e.g. wrong name given.



**ESLURM\_INVALID\_TIME\_VALUE** Invalid time value.

**ESLURM\_RESERVATION\_BUSY** Reservation is busy, e.g. trying to delete a reservation while in use.

**ESLURM\_RESERVATION\_NOT\_USABLE** Reservation not usable, e.g. trying to use an expired reservation.

## EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <slurm/slurm.h>
#include <slurm/slurm_errno.h>

int main (int argc, char *argv[])
{
    update_node_msg_t    update_node_msg;
    update_part_msg_t   update_part_msg;
    delete_part_msg_t   delete_part_msg;
    resv_desc_msg_t     resv_msg;
    char                 *resv_name = NULL;

    if (slurm_reconfigure ( ) ) {
        slurm_perror ("slurm_reconfigure error");
        exit (1);
    }

    slurm_init_part_desc_msg ( &update_part_msg );
    update_part_msg.name = "test.partition";
    update_part_msg.state_up = 0; /* partition down */
    if (slurm_create_partition (&update_part_msg)) {
        slurm_perror ("slurm_create_partition error");
        exit (1);
    }

    update_part_msg.state_up = 1; /* partition up */
    if (slurm_update_partition (&update_part_msg)) {
        slurm_perror ("slurm_update_partition error");
        exit (1);
    }

    delete_part_msg.name = "test.partition";
    if (slurm_delete_partition (&delete_part_msg)) {
        slurm_perror ("slurm_delete_partition error");
        exit (1);
    }

    slurm_init_update_node_msg (&update_node_msg);
    update_node_msg.node_names = "lx[10-12]";
    update_node_msg.node_state = NODE_STATE_DRAIN ;
    if (slurm_update_node (&update_node_msg)) {
        slurm_perror ("slurm_update_node error");
        exit (1);
    }

    slurm_init_resv_desc_msg ( &resv_msg );
    resv_msg.start_time = time(NULL) + 60*60; /* One hour from now */
    resv_msg.duration = 720; /* 12 hours/720 minutes */
    resv_msg.node_cnt = 10;
    resv_msg.accounts = "admin";
    resv_name = slurm_create_reservation (&resv_msg);
    if (!resv_name) {
        slurm_perror ("slurm_create_reservation error");
        exit (1);
    }
}
```



```
    free(resv_name);
    exit (0);
}
```

## NOTE

These functions are included in the libslurm library, which must be linked to your process for use (e.g. "cc -lslurm myprog.c").

## COPYING

Copyright (C) 2002–2007 The Regents of the University of California. Copyright (C) 2008–2010 Lawrence Livermore National Security. Produced at Lawrence Livermore National Laboratory (cf, DISCLAIMER). CODE–OCEC–09–009. All rights reserved.

This file is part of Slurm, a resource management program. For details, see <<https://slurm.schedmd.com/>>.

Slurm 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 of the License, or (at your option) any later version.

Slurm 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.

## SEE ALSO

[scontrol\(1\)](#), [slurm\\_get\\_errno\(3\)](#), [slurm\\_init\\_job\\_desc\\_msg\(3\)](#), [slurm\\_perror\(3\)](#), [slurm\\_strerror\(3\)](#), [slurm.conf\(5\)](#)

