NAME

slurm_step_ctx_create, slurm_step_ctx_create_no_alloc, slurm_step_ctx_daemon_per_node_hack, slurm_step_ctx_get, slurm_step_ctx_params_t_init, slurm_jobinfo_ctx_get, slurm_spawn_kill, slurm_step_ctx_destroy − Slurm task spawn functions

SYNTAX

#include <slurm/slurm.h>

slurm_step_ctx **slurm_step_ctx_create** (
slurm_step_ctx_params_t *_step_req_
);

slurm_step_ctx **slurm_step_ctx_create_no_alloc** (
slurm_step_ctx_params_t *_step_req_
);

int **slurm_step_ctx_daemon_per_node_hack** (
slurm_step_ctx_t *_ctx_
);

int **slurm_step_ctx_get** (
slurm_step_ctx_t *_ctx_,
int _ctx_key_,
...
);

int **slurm_jobinfo_ctx_get** (
switch_jobinfo_t _jobinfo_,
int _data_type_,
void *_data_
);

void **slurm_step_ctx_params_t_init** (
slurm_step_ctx_params_t *_step_req_
);

int **slurm_spawn** {
slurm_step_ctx _ctx_,
int *_fd_array_
);

int **slurm_spawn_kill** {
slurm_step_ctx _ctx_,
uint16_t _signal_
);

int **slurm_step_ctx_destroy** {
slurm_step_ctx _ctx_
);

ARGUMENTS

step_req
Specifies the pointer to the structure with job step request specification. See slurm.h for full details on the data structure’s contents.

ctx Job step context. Created by **slurm_step_ctx_create**, or **slurm_step_ctx_create_no_alloc** used in subsequent function calls, and destroyed by **slurm_step_ctx_destroy**.

ctx_key Identifies the fields in *ctx* to be collected by **slurm_step_ctx_get**.

data Storage location for requested data. See *data_type* below.

data_type
Switch−specific data requested. The interpretation of this field depends upon the switch plugin in use.



fd_array

Array of socket file descriptors to be connected to the initiated tasks. Tasks will be connected to these file descriptors in order of their task id. This socket will carry standard input, output and error for the task. *jobinfo* Switch−specific job information as returned by **slurm_step_ctx_get**.

signal Signal to be sent to the spawned tasks.

DESCRIPTION

slurm_jobinfo_ctx_get Get values from a *jobinfo* field as returned by **slurm_step_ctx_get**. The operation of this function is highly dependent upon the switch plugin in use.

slurm_step_ctx_create Create a job step context. To avoid memory leaks call **slurm_step_ctx_destroy** when the use of this context is finished. NOTE: this function creates a slurm job step. Call **slurm_spawn** in a timely fashion to avoid having job step credentials time out. If **slurm_spawn** is not used, explicitly cancel the job step.

slurm_step_ctx_create_no_alloc Same as above, only no allocation is made. To avoid memory leaks call **slurm_step_ctx_destroy** when the use of this context is finished.

slurm_step_ctx_daemon_per_node_hack Hack the step context to run a single process per node, regardless of the settings selected at slurm_step_ctx_create time.

slurm_step_ctx_get Get values from a job step context. *ctx_key* identifies the fields to be gathered from the job step context. Subsequent arguments to this function are dependent upon the value of *ctx_key*. See the **CONTEXT KEYS** section for details.

slurm_step_ctx_params_t_init This initializes parameters in the structure that you will pass to slurm_step_ctx_create().

slurm_spawn Spawn tasks based upon a job step context and establish communications with the tasks using the socket file descriptors specified. Note that this function can only be called once for each job step context. Establish a new job step context for each set of tasks to be spawned.

slurm_spawn_kill Signal the tasks spawned for this context by **slurm_spawn**.

slurm_step_ctx_destroy Destroy a job step context created by **slurm_step_ctx_create**.

CONEXT KEYS

SLURM_STEP_CTX_ARGS

Set the argument count and values for the executable. Accepts two additional arguments, the first of type int and the second of type char **.

SLURM_STEP_CTX_CHDIR

Have the remote process change directory to the specified location before beginning execution. Accepts one argument of type char * identifying the directory's pathname. By default the remote process will execute in the same directory pathname from which it is spawned. NOTE: This assumes that same directory pathname exists on the other nodes.

SLURM_STEP_CTX_ENV

Sets the environment variable count and values for the executable. Accepts two additional arguments, the first of type int and the second of type char **. By default the current environment variables are copied to started task's environment.

SLURM_STEP_CTX_RESP

Get the job step response message. Accepts one additional argument of type job_step_create_response_msg_t **.

SLURM_STEP_CTX_STEPID

Get the step id of the created job step. Accepts one additional argument of type uint32_t *.

SLURM_STEP_CTX_TASKS

Get the number of tasks per node for a given job. Accepts one additional argument of type uint32_t **. This argument will be set to point to an array with the task counts of each node in an element of the array. See **SLURM_STEP_CTX_TID** below to determine the task ID numbers associated with each of those tasks.



SLURM_STEP_CTX_TID
Get the task ID numbers associated with the tasks allocated to a specific node. Accepts two additional arguments, the first of type int and the second of type uint32_t **. The first argument identifies the node number of interest (zero origin). The second argument will be set to point to an array with the task ID numbers of each task allocated to the node (also zero origin). See **SLURM_STEP_CTX_TASKS** above to determine how many tasks are associated with each node.

RETURN VALUE

For **slurm_step_ctx_create** a context is return upon success. On error NULL is returned and the Slurm error code is set appropriately.

For all other functions zero is returned upon success. On error, −1 is returned, and the Slurm error code is set appropriately.

ERRORS

EINVAL Invalid argument

SLURM_PROTOCOL_VERSION_ERROR Protocol version has changed, re−link your code.

ESLURM_INVALID_JOB_ID the requested job id does not exist.

ESLURM_ALREADY_DONE the specified job has already completed and can not be modified.

ESLURM_ACCESS_DENIED the requesting user lacks authorization for the requested action (e.g. trying to delete or modify another user's job).

ESLURM_DISABLED the ability to create a job step is currently disabled. This is indicative of the job being suspended. Retry the call as desired.

ESLURM_INTERCONNECT_FAILURE failed to configure the node interconnect.

ESLURM_BAD_DIST task distribution specification is invalid.

SLURM_PROTOCOL_SOCKET_IMPL_TIMEOUT Timeout in communicating with Slurm controller.

EXAMPLE

SEE **slurm_step_launch**(3) man page for an example of slurm_step_ctx_create and slurm_step_launch in use together.

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) 2004-2007 The Regents of the University of California. 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

slurm_allocate_resources(3), **slurm_job_step_create**(3), **slurm_kill_job**(3), **slurm_get_errno**(3), **slurm_perror**(3), **slurm_strerror**(3), **srun**(1)

