Sun™ Grid Engine 5.3 and Sun™ Grid Engine, Enterprise Edition 5.3 Reference Manual

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 816-4767-10
April 2002, Revision 01

Send comments about this document to: `docfeedback@sun.com`

Contents

Tables

Preface

The *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* is a compilation of manual (`man`) pages that accompany the two products. This manual includes information that is mostly common to both products. However, when the information is specific to one product only, that is noted.

How This Book Is Organized

In addition to this preface and the table of contents, this manual consists of one chapter only.

- Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Manual Pages

Using UNIX Commands

This document does not contain information on basic UNIX® commands and procedures, such as shutting down the system, booting the system, and configuring devices.

See one or more of the following for this information.

- *Solaris Handbook for Sun Peripherals*
- AnswerBook2™ online documentation for the Solaris™ operating environment
- Other software documentation that you received with your system

Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123`	What you type, when contrasted with on-screen computer output	`%` **`su`**
		`Password:`

AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type `rm` *filename*.

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell superuser	#

Related Documentation

Application	Title	Part Number
Administration and use	*Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide*	816-4739-10
Administration and use	*Sun Grid Engine 5.3 Administration and User's Guide*	816-2077-11

Accessing Sun Documentation Online

A broad selection of Sun system documentation is located at:

`http://www.sun.com/products-n-solutions/hardware/docs`

A complete set of Solaris documentation and many other titles are located at:

`http://docs.sun.com`

At that site, you will also find information on how to order *printed* copies of this guide.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can email your comments to Sun at:

`docfeedback@sun.com`

Please include the part number (816-4767-10) of your document in the subject line of your email.

Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Manual Pages

Introduction

This document contains the manual pages as included in the Sun™ Grid Engine distribution. Note that several of the commands listed in this document apply only to the Sun Grid Engine, Enterprise Edition 5.3 product, as indicated in the descriptions of such commands.

sge_intro(1)

Name

Sun Grid Engine Introduction – a facility for executing UNIX jobs on remote machines

Description

Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 software—both of which will be referred to as Sun Grid Engine throughout most of the descriptions in this manual—are each products for executing UNIX batch jobs (shell scripts) on a pool of cooperating workstations. Jobs are queued and executed remotely on workstations at times when those workstations would otherwise be idle or only lightly loaded. The work load is distributed among the workstations in the cluster corresponding to the load situation of each machine and the resource requirements of the jobs.

User level checkpointing programs are supported and a transparent checkpointing mechanism is provided (see `sge_ckpt(1)`). Checkpointing jobs migrate from workstation to workstation without user intervention on load demand. In addition to batch jobs, interactive jobs and parallel jobs can also be submitted to Sun Grid Engine.

User Interface

The Sun Grid Engine user interface consists of several programs, which are described separately in TABLE 1.

TABLE 1 User Interface Programs and Descriptions

Program Name	**Description**
`qacct(1)`	Extracts arbitrary accounting information from the cluster logfile.
`qalter(1)`	Changes the characteristics of already submitted jobs.
`qconf(1)`	Provides the user interface for configuring, modifying, deleting and querying queues and the cluster configuration.
`qdel(1)`	Provides the means for a user/operator/manager to cancel jobs.
`qhold(1)`	Holds back submitted jobs from execution.
`qhost(1)`	Displays status information about Sun Grid Engine execution hosts.
`qlogin(1)`	Initiates a telnet or similar login session with automatic selection of a low loaded and suitable host.
`qmake(1)`	Replacement for the standard UNIX `make` facility. It extends make by its ability to distribute independent `make` steps across a cluster of suitable machines.
`qmod(1)`	Allows the owner(s) of a queue to suspend and enable all queues associated with his machine (all currently active processes in this queue are also signaled) or to suspend and enable jobs executing in the owned queues.
`qmon(1)`	Provides a Motif command interface to all Sun Grid Engine functions. The status of all or a private selection of the configured queues is displayed on-line by changing colors at corresponding queue icons.
`qresub(1)`	Creates new jobs by copying currently running or pending jobs.

TABLE 1 User Interface Programs and Descriptions *(Continued)*

Program Name	Description
`qrls(1)`	Releases holds from jobs previously assigned to them e.g. via `qhold(1)` (see above).
`qrsh(1)`	Can be used for various purposes such as providing remote execution of interactive applications via Sun Grid Engine comparable to the standard UNIX facility `rsh`, to allow for the submission of batch jobs which, upon execution, support terminal I/O (standard/error output and standard input) and terminal control, to provide a batch job submission client which remains active until the job has finished or to allow for the Sun Grid Engine-controlled remote execution of the tasks of parallel jobs.
`qselect(1)`	Prints a list of queue names corresponding to specified selection criteria. The output of qselect is usually fed into other Sun Grid Engine commands to apply actions on a selected set of queues.
`qsh(1)`	Opens an interactive shell (in an `xterm(1)`) on a low loaded host. Any kind of interactive jobs can be run in this shell.
`qstat(1)`	Provides a status listing of all jobs and queues associated with the cluster.
`qsub(1)`	The user interface for submitting a Sun Grid Engine job
`qtcsh(1)`	Fully compatible replacement for the widely known and used UNIX C-Shell (`csh`) derivative `tcsh`. It provides a command-shell with the extension of transparently distributing execution of designated applications to suitable and lightly loaded hosts via Sun Grid Engine.

See Also

`sge_ckpt(1)`, `qacct(1)`, `qalter(1)`, `qconf(1)`, `qdel(1)`, `qhold(1)`, `qhost(l)`, `qlogin(1)`, `qmake(1)`, `qmod(1)`, `qmon(1)`, `qresub(1)`, `qrls(1)`, `qrsh(1)`, `qselect(1)`, `qsh(1)`, `qstat(1)`, `qsub(1)`, `qtcsh(1)`, *Sun Grid Engine 5.3 Administration and User's Guide, and Sun Grid Engine Enterprise Edition 5.3 Administration and User's Guide*

Copyright

sge_ckpt(1)

Name

Sun Grid Engine Checkpointing – the Sun Grid Engine checkpointing mechanism and checkpointing support

Description

Sun Grid Engine supports two levels of checkpointing: the user level and a operating system provided transparent level. User level checkpointing refers to applications, which do their own checkpointing by writing restart files at certain times or algorithmic steps and by properly processing these restart files when restarted.

Transparent checkpointing has to be provided by the operating system and is usually integrated in the operating system kernel. An example for a kernel integrated checkpointing facility is the Hibernator package from Softway for SGI IRIX platforms.

Checkpointing jobs need to be identified to the Sun Grid Engine system by using the `-ckpt` option of the `qsub(1)` command. The argument to this flag refers to a so called checkpointing environment, which defines the attributes of the checkpointing method to be used (see `checkpoint(5)` for details). Checkpointing environments are setup by the `qconf(1)` options `-ackpt`, `-dckpt`, `-mckpt` and `-sckpt`. The `qsub(1)` option `-c` can be used to overwrite the `when` attribute for the referenced checkpointing environment.

If a queue is of the type *CHECKPOINTING*, jobs need to have the checkpointing attribute flagged (see the `-ckpt` option to `qsub(1)`) to be permitted to run in such a queue. As opposed to the behavior for regular batch jobs, checkpointing jobs are aborted under conditions for which batch or interactive jobs are suspended or even stay unaffected. These conditions are:

- Explicit suspension of the queue or job via `qmod(1)` by the cluster administration or a queue owner if the `x` occasion specifier (see `qsub(1)-c` and `checkpoint(5)`) was assigned to the job.
- A load average value exceeding the migration threshold as configured for the corresponding queues (see `queue_conf(5)`).
- Shutdown of the Sun Grid Engine execution daemon `sge_execd(8)` being responsible for the checkpointing job.

After aborting, the jobs will migrate to other queues unless they were submitted to one specific queue by an explicit user request. The migration of jobs leads to a dynamic load balancing.

Note – Aborting checkpointed jobs will free all resources (memory, swap space) which the job occupies at that time. This is opposed to the situation for suspended regular jobs, which still cover swap space.

Restrictions

When a job migrates to a queue on another machine at present no files are transferred automatically to that machine. This means that all files which are used throughout the entire job including restart files, executables and scratch files must be visible or transferred explicitly (e.g. at the beginning of the job script).

There are also some practical limitations regarding use of disk space for transparently checkpointing jobs. Checkpoints of a transparently checkpointed application are usually stored in a checkpoint file or directory by the operating system. The file or directory contains all the text, data, and stack space for the process, along with some additional control information. This means jobs which use a very large virtual address space will generate very large checkpoint files. Also the workstations on which the jobs will actually execute may have little free disk space. Thus it is not always possible to transfer a transparent checkpointing job to a machine, even though that machine is idle. Since large virtual memory jobs must wait for a machine that is both idle, and has a sufficient amount of free disk space, such jobs may suffer long turnaround times.

See Also

`sge_intro(1)`, `qconf(1)`, `qmod(1)`, `qsub(1)`, `checkpoint(5)`*, Sun Grid Engine 5.3 Administration and User's Guide, and Sun Grid Engine Enterprise Edition 5.3 Administration and User's Guide*

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qacct(1)

Name

`qacct` – Report and account for Sun Grid Engine usage.

Synopsis

`qacct` [`-A` *Account*] [`-b` *BeginTime*] [`-d` *Days*]
[`-e` *EndTime*] [`-g` [*GroupId*|*GroupName*]]
[`-h` [*HostName*]] [`-help`] [`-history` *HistoryPath*]
[`-j` [*JobId*|*JobName*]] [`-l` *attr=val,...*] [`-nohist`]
[`-o` [*Owner*]] [`-pe` [*PEname*]] [`-q` [*Q_name*]]
[`-slots` [*SlotNumber*]] [`-t` *task_id_range_list*]
[`-P` [*Project*]] [`-D` [*Department*]] [`-f` *Account FileName*]

Description

The `qacct` utility scans the accounting data file (see `accounting(5)`) and produces a summary of information for wall-clock time, cpu-time, and system time for the categories of hostname, queue-name, group-name, owner-name, job-name, job-ID and for the queues meeting the resource requirements as specified with the `-l` switch. Combinations of each category are permitted. Alternatively, all or specific jobs can be listed with the `-j` switch. For example the search criteria could include summarizing for a queue and an owner, but not for two queues in the same request.

Options

TABLE 2 lists options to `qacct`.

TABLE 2 `qacct` Options

Option	Description
`-A` *Account*	The account for jobs to be summarized.
`-b` *BeginTime*	The earliest start time for jobs to be summarized, in the format *[[CC]YY]MMDDhhmm[.SS]*. See also `-d` option.
`-d` *Days*	The number of days to summarize and print accounting information on. If used together with the `-b` *BeginTime* option (see above), jobs started within *BeginTime* to *BeginTime* + *Days* are counted. If used together with the `-e` *EndTime* (see below) option, count starts at *EndTime* - *Days*.
`-e` *EndTime*	The latest start time for jobs to be summarized, in the format *[[CC]YY]MMDDhhmm[.SS]*. See also `-d` option.
[`-f` *AcctFileName*]	The accounting file to be used. If omitted, the system default accounting file is processed.

TABLE 2 `qacct` Options *(Continued)*

Option	Description
`-g [`*GroupId*`|`*GroupName*`]`	The numeric system group id or the group alphanumeric name of the job owners to be included in the accounting. If *GroupId/GroupName* is omitted, all groups are accounted.
`-h` [*HostName*]	The case-insensitive name of the host upon which accounting information is requested. If the name is omitted, totals for each host are listed separately.
`-help`	Display help information for the qacct command.
`-history` *HistoryPath*	The directory path where the historical queue and complexes configuration data is located, which is used for resource requirement matching in conjunction with the `-l` switch. If the latter is not set, this option is ignored.
`-j [[`*JobName*`|`*JobId*`]]`	The name or ID of the job during execution for which accounting information is printed. If neither a name nor an ID is given all jobs are enlisted. This option changes the output format of `qacct`. If activated, CPU times are no longer accumulated but the "raw" accounting information is printed in a formatted form instead. See `accounting(5)` for an explanation of the displayed information.
`-l` *attr=val,...*	A resource requirement specification which must be met by the queues in which the jobs being accounted were executing. The matching is performed with historical data, i.e. it reflects the situation of the queue and complexes configuration at the time of the job start. The resource request is very similar to the one described in `qsub(1)`. The main difference is that ever changing load information may not be requested as it is not contained in the historical configuration data being used.
`-nohist`	Only useful together with the `-l` option. It forces `qacct` not to use historical queue and complexes configuration data for resource requirement matching but instead retrieve actual queue and complexes configuration from `sge_qmaster(8)`. Note that this may lead to confusing statistical results, as the current queue and complexes configuration may differ significantly from the situation being valid for past jobs. Also note that all hosts being referenced in the accounting file have to be up and running in order to get results.
`-o` [*Owner*]	The name of the owner of the jobs for which accounting statistics are assembled. If the optional Owner argument is omitted, a listing of the accounting statistics of all job owners being present in the accounting file is produced.

TABLE 2 `qacct` Options *(Continued)*

Option	Description
`-pe` [*PEname*]	The name of the parallel environment for which usage is summarized. If *PEname* is not given, accounting data is listed for each parallel environment separately.
`-q` [*Q_name*]	The name of the queue for which usage is summarized. If *Q_name* is not given, accounting data is listed for each queue separately.
`-slots [`*SlotNumber*`]`	The number of queue slots for which usage is summarized. If *SlotNumber* is not given, accounting data is listed for each number of queue slots separately.
`-t` *task_id_range_list*	Only available together with the `-j` option described above. The `-t` switch specifies the job array task range, for which accounting information should be printed. Syntax and semantics of *task_id_range_list* are identical to that one described under the `-t` option to `qsub(1)`. See there also for further information on job arrays.
`-P` [*Project*]	The name of the project for which usage is summarized. If *Project* is not given, accounting data is listed for each owner project separately. Projects are only used when running in Sun Grid Engine, Enterprise Edition mode.
`-D` [*Department*]	The name of the department for which usage is summarized. If *Department* is not given, accounting data is listed for each owner department separately. Departments are only used when running in Sun Grid Engine, Enterprise Edition mode

Environment Variables

TABLE 3 describes the environment variables associated with `qacct`.

TABLE 3 `qacct` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qacct` uses (in the order of precedence): • The name of the cell specified in the environment variable, `SGE_CELL`, if it is set • The name of the default cell; e.g., `default`
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qacct` client resides. Per default the local host is used.

Files

- Sun Grid Engine default accounting file – *<sge_root>*/*<cell>*`/common/accounting`
- Sun Grid Engine default history database – *<sge_root>*/*<cell>*`/common/history`
- Sun Grid Engine master host file – *<sge_root>*/*<cell>*`/common/act_qmaster`

See Also

`sge_intro(1)`, `qsub(1)`, `accounting(5)`, `sge_qmaster(8)`, `sge_commd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qconf(1)

Name

`qconf` – Sun Grid Engine Queue Configuration

Description

`qconf` allows the system administrator to add, delete, and modify the current Sun Grid Engine configuration, including queue management, host management, complex management and user management. `qconf` also allows you to examine the current queue configuration for existing queues.

Options

Unless noted otherwise, the options listed in TABLE 4 and the corresponding operations are available to all users with a valid account.

TABLE 4 `qconf` Options

Option	Description
`-Aattr` *obj_spec fname obj_instance,...*	*Add to object attributes* – Similar to `-aattr` (see below) but takes specifications for the object attributes to be enhanced from file named fname. As opposed to `-aattr`, multiple attributes can be enhanced. Their specification has to be enlisted in *fname* following the file format of the corresponding object (see `queue_conf(5)` for the queue, for example). This option requires root/manager privileges.
`-Ac` *complex_name fname*	*Add complex* – Add the complex *complex_name* defined in fname to the Sun Grid Engine cluster. The format of the complex specification is described in `complex(5)`. Requires root or manager privileges.
`-Acal` *fname*	*Add calendar* – Adds a new calendar definition to the Sun Grid Engine environment. Calendars are used in Sun Grid Engine for defining availability and unavailability schedules of queues. The format of a calendar definition is described in `calendar_conf(5)`. The calendar definition is taken from the file *fname*. Requires root/ manager privileges.
`-Ackpt` *fname*	*Add checkpoint environment* – Add the checkpointing environment as defined in *fname* (see `checkpoint(5)`) to the list of supported checkpointing environments. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–Aconf` *file_list*	*Add configurations* – Add the cluster configurations (see `sge_conf(5)`) specified in the files enlisted in the comma-separated *file_list*. Requires root or manager privileges.
`–Ae` *fname*	*Add execution host* – Add the execution host defined in *fname* to the Sun Grid Engine cluster. The format of the execution host specification is described in `host_conf(5)`. Requires root or manager privileges.
`–Ap` *fname*	*Add PE configuration* – Add the parallel environment (PE) defined in *fname* to the Sun Grid Engine cluster. Requires root or manager privileges.
`–Aprj` *fname*	*Add new project* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Adds the project description defined in *fname* to the list of registered projects (see `project(5)`). Requires root or manager privileges.
`–Aq` *fname*	*Add new queue* – Add the queue defined in *fname* to the Sun Grid Engine cluster. Requires root or manager privileges.
`–Au` *fname*	*Add ACL* – Adds a user access list (ACL) to Sun Grid Engine. User lists are used for queue usage authentication. Requires root/manager/operator privileges.
`–Dattr` *obj_spec fname obj_instance,...*	*Delete from object attributes* – Similar to `-dattr` (see below) but the definition of the list attributes from which entries are to be deleted is contained in the file named *fname*. As opposed to `-dattr`, multiple attributes can be modified. Their specification has to be enlisted in *fname* following the file format of the corresponding object (see `queue_conf(5)` for the queue, for example). Requires root/manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–Mattr` *obj_spec fname obj_instance,...*	*Modify object attributes* – Similar to `-mattr` (see below) but takes specifications for the object attributes to be modified from file named *fname*. As opposed to `-mattr`, multiple attributes can be modified. Their specification has to be enlisted in *fname* following the file format of the corresponding object (see `queue_conf(5)` for the queue, for example). Requires root/manager privileges.
`–Mc` *complex_name fname*	*Modify complex* – Overwrites the specified complex by the contents of *fname*. The argument file must comply to the format specified in `complex(5)`. Requires root or manager privileges.
`–Mcal` *fname*	Modify calendar – Overwrites the calendar definition as specified in *fname*. The argument file must comply to the format described in `calendar_conf(5)`. Requires root or manager privileges.
`–Mckpt` *fname*	*Modify checkpoint environment* – Overwrite an existing checkpointing environment with the definitions in *fname* (see `checkpoint(5)`). The name attribute in *fname* has to match an existing checkpointing environment. Requires root or manager privileges.
`–Me` *fname*	*Modify execution host* – Overwrites the execution host configuration for the specified host with the contents of *fname*, which must comply to the format defines in `host_conf(5)`. Requires root or manager privileges.
`–Mp` *fname*	*Modify PE configuration* – Same as `–mp` (see below) but instead of invoking an editor to modify the PE configuration the file *fname* is considered to contain a changed configuration. Refer to `sge_pe(5)` for details on the PE configuration format. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–Mprj` *fname*	*Modify project configuration* – Same as `–mprj` (see below) but instead of invoking an editor to modify the project configuration the file *fname* is considered to contain a changed configuration. Refer to `project(5)` for details on the project configuration format. Requires root or manager privileges.
`–Mq` *fname*	*Modify queue configuration* – Same as `–mq` (see below) but instead of invoking an editor to modify the queue configuration the file *fname* is considered to contain a changed configuration. Refer to `queue_conf(5)` for details on the queue configuration format. Requires root or manager privileges.
`–Mqattr` *fname q_name,...* This command is deprecated. Use `-Mattr` instead.	*Modify queue attributes* – Allows changing of selected queue configuration attributes in multiple queues with a single command. In all queues contained in the comma separated queue name list the queue attribute definitions contained in *fname* will be applied. Queue attributes not contained in *fname* will be left unchanged. All queue attributes can be modified except for *qname* and *qhostname*. Refer to `queue_conf(5)` for details on the queue configuration format. Requires root or manager privileges.
`–Mu` *fname*	*Modify ACL* – Takes the user access list (ACL) defined in *fname* to overwrite any existing ACL with the same name. See `access_list(5)` for information on the ACL configuration format. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–Rattr` *obj_spec fname obj_instance,...*	*Replace object attributions* – Similar to `-rattr` (see below) but the definition of the list attributes whose content is to be replace is contained in the file named *fname*. As opposed to `-rattr`, multiple attributes can be modified. Their specification has to be enlisted in *fname* following the file format of the corresponding object (see `queue_conf(5)` for the queue, for example). Requires root/manager privileges.
`–aattr` *obj_spec attr_name val obj_instance,...*	*Add to object attributes* – Allows adding specifications to a single configuration list attribute in multiple instances of an object with a single command. Currently supported objects are the queue and the host configuration being specified as *queue* or *host* in *obj_spec*. The queue *load_thesholds* parameter is an example of a list attribute. With the `-aattr` option, entries can be added to such lists, while they can be deleted with `-dattr`, modified with `-mattr`, and replaced with `-rattr`. The name of the configuration attribute to be enhanced is specified with *attr_name* followed by *val* as a *name=value* pair. The comma separated list of object instances (e.g., the list of queues) to which the changes have to be applied are specified at the end of the command. The following restriction applies: For the *host* object, the *load_values* attribute cannot be modified (see `host_conf(5)`). Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-ac` *complex_name*	*Add complex* – Adds a complex to the Sun Grid Engine environment. Complex entries contain one or more resources which may be requested by jobs submitted to the system. The `complex(5)` manual page contains detailed information about the format of a complex definition. When using the `-ac` option the complex name is given in the command option. `qconf` will then open a temporary file and start up the text editor indicated by the environment variable `EDITOR` (default editor is `vi(1)` if `EDITOR` is not set). After entering the complex definition and closing the editor the new complex is checked and registered with `sge_qmaster(8)`. Requires root/manager privileges.
`-acal` *calendar_name*	*Add calendar* – Adds a new calendar definition to the Sun Grid Engine environment. Calendars are used in Sun Grid Engine for defining availability and unavailability schedules of queues. The format of a calendar definition is described in `calendar_conf(5)`. With the calendar name given in the option argument `qconf` will open a temporary file and start up the text editor indicated by the environment variable `EDITOR` (default editor is `vi(1)` if `EDITOR` is not set). After entering the calendar definition and closing the editor the new calendar is checked and registered with `sge_qmaster(8)`. Requires root/manager privileges.

TABLE 4 qconf Options *(Continued)*

Option	Description
-ackpt *ckpt_name*	*Add checkpoint environment* – Adds a checkpointing environment under the name *ckpt_name* to the list of checkpointing environments maintained by Sun Grid Engine and to be usable to submit checkpointing jobs (see checkpoint(5) for details on the format of a checkpointing environment definition). qconf retrieves a default checkpointing environment configuration and executes vi(1*)* (or $EDITOR if the EDITOR environment variable is set) to allow you to customize the checkpointing environment configuration. Upon exit from the editor, the checkpointing environment is registered with sge_qmaster(8). Requires root/manager privileges.
-aconf *host,...*	A*dd configuration* – Successively adds cluster configurations (see sge_conf(5)) For the hosts in the comma-separated *file_list*. For each host, an editor ($EDITOR indicated or vi(1)) is invoked and the configuration for the host can be entered. The configuration is registered with sge_qmaster(8) after saving the file and quitting the editor. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	**Description**
`-ae` *[host_template]*	*Add execution host* – Adds a host to the list of Sun Grid Engine execution hosts. If a queue is configured on a host this host is automatically added to the Sun Grid Engine execution host list. Adding execution hosts explicitly offers the advantage to be able to specify parameters like load scale values with the registration of the execution host. However, these parameters can be modified (from their defaults) at any later time via the `-me` option described below. If the *host_template* argument is present, `qconf` retrieves the configuration of the specified execution host from `sge_qmaster(8)` or a generic template otherwise. The template is then stored in a file and `qconf` executes `vi(1)` (or the editor indicated by `$EDITOR` if the `EDITOR` environment variable is set) to change the entries in the file. The format of the execution host specification is described in `host_conf(5)`. When the changes are saved in the editor and the editor is quit the new execution host is registered with `sge_qmaster(8)`. This option requires root/manager privileges.
`-ah` *hostname,...*	*Add administrative host* – Adds hosts *hostname* to the Sun Grid Engine trusted host list (a host must be in this list to execute administrative Sun Grid Engine commands, the sole exception to this being the execution of `qconf` on the `sge_qmaster(8)` node). The default Sun Grid Engine installation procedures usually add all designated execution hosts (see the `-ae` option above) to the Sun Grid Engine trusted host list automatically. Requires root or manager privileges.
`-am` *user,...*	*Add managers* – Adds the indicated users to the Sun Grid Engine manager list. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-ao` *user,...*	*Add operator* – Adds the indicated users to the Sun Grid Engine operator list. Requires root/manager privileges.
`-ap` *pe_name*	*Add new PE* – Adds a *Parallel Environment* (PE) description under the name *pe_name* to the list of PEs maintained by Sun Grid Engine and to be usable to submit parallel jobs (see `sge_pe(5)` for details on the format of a PE definition). `qconf` retrieves a default PE configuration and executes `vi(1)` (or `$EDITOR` if the `EDITOR` environment variable is set) to allow you to customize the PE configuration. Upon exit from the editor, the PE is registered with `sge_qmaster(8)`. Requires root/manager privileges.
`-aprj`	*Add new project* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Adds a project description to the list of registered projects (see `project(5)`). `qconf` retrieves a template project configuration and executes `vi(1)` (or `$EDITOR` if the `EDITOR` environment variable is set) to allow you to customize the new project. Upon exit from the editor, the template is registered with `sge_qmaster(8)`. Requires root or manager privileges.
`-aq [`*q_template*`]`	*Add new queue* – `qconf` retrieves either the default queue configuration (see `queue_conf(5)`) or the configuration of the queue *q_template* (if the optional argument is present) and executes `vi(1)` (or `$EDITOR` if the `EDITOR` environment variable is set) to allow you to customize the queue configuration. Upon exit from the editor, the queue is registered with `sge_qmaster(8)`. A minimal configuration requires only that the queue name and queue host name be set. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-as` *hostname,...*	*Add submit hosts* – Add hosts *hostname* to the list of hosts allowed to submit Sun Grid Engine jobs and control their behavior only. Requires root or manager privileges.
`-astnode` *node_path=shares,...*	*Add share tree node* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Adds the specified share tree node(s) to the share tree (see `share_tree(5)`). The *node_path* is a hierarchical path (`[/]`*node_name*`[[/.]`*node_name...]*) specifying the location of the new node in the share tree. The base name of the *node_path* is the name of the new node. The node is initialized to the number of specified shares. Requires root or manager privileges.
`-astree`	*Add share tree* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Adds the definition of a share tree to the system (see `share_tree(5)`). A template share tree is retrieved and an editor (either `vi(1`*)* or the editor indicated by `$EDITOR`) is invoked for modifying the share tree definition. Upon exiting the editor, the modified data is registered with `sge_qmaster(8)`. Requires root or manager privileges.
`-au` *user,... acl_name,...*	*Add users to ACLs* – Adds users to Sun Grid Engine user access lists (ACLs). User lists are used for queue usage authentication. Requires root/manager/operator privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-Auser` *fname*	*Add user* – (This option is supported only for a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems.) Add the user defined in *fname* to the Sun Grid Engine, Enterprise Edition cluster. The format of the user specification is described in `user(5)`. Requires root or manager privileges.
`–auser`	*Add user* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Adds a user to the list of registered users (see `user(5)`). This command invokes an editor (either `vi(1)` or the editor indicated by the `EDITOR` environment variable) for a template user. The new user is registered after changing the entry and exiting the editor. Requires root or manager privileges.
`-clearusage`	*Clear sharetree usage* – (This option is supported only for a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems.) Clears all user and project usage from the sharetree. All usage will be initialized back to zero.
`–cq` *queue_name,...*	*Clean queue* – Cleans queue from jobs which haven't been reaped. Primarily a development tool. Requires root/manager/operator privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-dattr` *obj_spec attr_name val obj_instance,...*	*Delete in-object attributes* – Allows deleting specifications in a single configuration list attribute in multiple instances of an object with a single command. Currently supported objects are the queue and the host configuration being specified as *queue* or *host* in *obj_spec*. The queue *load_thesholds* parameter is an example of a list attribute. With the `-dattr` option, entries can be deleted from such lists, while they can be added with `-aattr`, modified with `-mattr`, and replaced with `-rattr`. The name of the configuration attribute to be modified is specified with *attr_name* followed by *val* defining the name of the attribute list entry to be deleted. The comma-separated list of object instances (e.g., the list of queues) to which the changes have to be applied are specified at the end of the command. The following restriction applies: For the *host* object, the *load_values* attribute cannot be modified (see `host_conf(5)`). Requires root or manager privilege.
`-dc` *complex_name,...*	*Delete complex* – Deletes complexes from Sun Grid Engine. this option requires root/manager privileges.
`-dcal` *calendar_name,...*	*Delete calendar* – Deletes the specified calendar definition from Sun Grid Engine. Requires root/manager privileges.
`-dckpt` *ckpt_name*	*Delete checkpoint environment* – Deletes the specified checkpointing environment. Requires root/manager privileges.
`-dconf` *host,...*	*Delete configuration* – The configuration entry for the specified hosts is deleted from the configuration list. Requires root or manager privilege.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-de` *host_name,...*	*Delete execution host* – Deletes hosts from the Sun Grid Engine execution host list. Requires root/manager privileges.
`-dh` *host_name,...*	*Delete administrative host* – Deletes hosts from the Sun Grid Engine trusted host list. The host on which `sge_qmaster(8)` is currently running cannot be removed from the list of administrative hosts. Thisoption requires root/manager privileges.
`-dm` *user[,user,...]*	*Delete managers* – Deletes managers from the manager list. Requires root/manager privileges.
`-do` *user[,user,...]*	*Delete operators* – Deletes operators from the operator list. Requires root/manager privileges.
`-dp` *pe_name*	*Delete parallel environment* – Deletes the specified parallel environment (PE). Requires root/manager privileges.
`-dprj` *project,...*	*Delete projects* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Deletes the specified project(s). Requires root/manager privileges.
`-dq` *queue_name,...*	*Delete queue* – Removes the specified queue(s). Active jobs will be allowed to run to completion. Requires root/manager privileges.
`-ds` *host_name,...*	*Delete submit host* – Deletes hosts from the Sun Grid Engine submit host list. Requires root/manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–dstnode` *node_path,...*	*Delete share tree node* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Deletes the specified share tree node(s). The *node_path* is a hierarchical path (*[/]node_name[[/.]node_name...]*) specifying the location of the node to be deleted in the share tree. Requires root or manager privileges.
`–dstree`	*Delete share tree* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Deletes the current share tree. Requires root or manager privileges.
`–du` *user,... acl_name,...*	*Delete users from ACL* – Deletes one or more users from one or more Sun Grid Engine user access lists (ACLs). Requires root/manager/operator privileges.
`–dul` *acl_name,...*	*Delete user lists* – Deletes one or more user lists from the system. Requires root/manager/operator privileges.
`–duser` *user,...*	*Delete users* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Deletes the specified user(s) from the list of registered users. Requires root or manager privileges.
`–help`	Prints a listing of all options.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-k{m|s|e[j] {`*host,...*`|all}}}`	*Shutdown Sun Grid Engine software* – Used to shut down Sun Grid Engine components (daemons). In the form `-km` `sge_qmaster(8)` is forced to terminate in a controlled fashion. In the same way the `-ks` switch causes termination of `sge_schedd(8)`. Shutdown of running `sge_execd(8)` processes currently registered is initiated by the `-ke` option. If `-kej` is specified instead, all jobs running on the execution hosts are aborted prior to termination of the corresponding `sge_execd(8`*)*. The comma-separated host list specifies the execution hosts to be addressed by the `–ke` and `-kej` option. If the keyword, `all`, is specified instead of a host list, all running `sge_execd(8)` processes are shut down. Requires root or manager privileges.
`-kec` *{id,...*`|all`*}*	*Kill event client* – Used to shut down event clients registered at `sge_qmaster(8)`. The comma-separated event client list specifies the event clients to be addressed by the `-kec` option. If the keyword, `all`, is specified instead of an event client list, all running event clients except special clients like the `sge_schedd(8)` are terminated. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	**Description**
`-mattr` *obj_spec attr_name val obj_instance,...*	*Modify object attributes* – Allows changing a single configuration attribute in multiple instances of an object with a single command. Currently supported objects are the queue and the host configuration being specified as *queue* or host in *obj_spec*. Note, that `-mattr` *queue attr_name val q_name, ...* is equivalent to `-mqattr` *attr_name val q_name,...* (see below). The latter is available for backward compatibility. The name of the configuration attribute to be modified is specified with *attr_name* followed by the value to which the attribute is going to be set. If the attribute is a list, such as the queue *load_thresholds*, *val* can be a *name=value* pair, in which case only a corresponding entry in the list is changed. Refer to the `-aattr`, `-dattr`, and `-rattr` options for a description of further means to change specifically such list attributes. The comma-separated list of object instances (e.g., the list of queues) to which the changes have to be applied are specified at the end of the command. The following restrictions apply: For the queue object, the *qname* and *qhostname* attributes cannot be modified (see `queue_conf(5)`). For the host object, the *hostname*, *load_values*, and *processors* attributes cannot be modified (see `host_conf(5)`). Requires root or manager privileges.
`-mc` *complex_name*	*Modify complex* – The specified complex configuration (see `complex(5)`) is retrieved, an editor is executed (either `vi(1)` or the editor indicated by `$EDITOR`) and the changed complex configuration is registered with `sge_qmaster(8)` upon exit of the editor. Requires root or manager privilege.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-mcal` *calendar_name*	*Modify calendar* – The specified calendar definition (see `calendar_conf(5)`) is retrieved, an editor is executed (either `vi(1)` or the editor indicated by `$EDITOR`) and the changed calendar definition is registered with `sge_qmaster(8)` upon exit of the editor. Requires root or manager privilege.
`-mckpt` *ckpt_name*	*Modify checkpoint. environment* – Retrieves the current configuration for the specified checkpointing environment, executes an editor (either `vi(1)` or the editor indicated by the `$EDITOR` environment variable) and registers the new configuration with the `sge_qmaster(8)`. Refer to `checkpoint(5)` for details on the checkpointing environment configuration format. Requires root or manager privileges.
`-mconf [`*host,...*`|`*global*`]`	*Modify configuration* – The configuration for the specified host is retrieved, an editor is executed (either `vi(1)` or the editor indicated by `$EDITOR`) and the changed configuration is registered with `sge_qmaster(8)` upon exit of the editor. If the optional host argument is omitted or if the special host name, `global` is specified, the cell global configuration is modified. The format of the host configuration is described in `sge_conf(5)`. Requires root or manager privileges.
`-me` *hostname*	*Modify execution host* – Retrieves the current configuration for the specified execution host, executes an editor (either `vi(1)` or the editor indicated by the `$EDITOR` environment variable) and registers the changed configuration with `sge_qmaster(8)` upon exit from the editor. The format of the execution host configuration is described in `host_conf(5)`. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-mp` *pe_name*	*Modify PE configuration* – Retrieves the current configuration for the specified *parallel environment* (PE), executes an editor (either `vi(1)` or the editor indicated by the `EDITOR` environment variable) and registers the new configuration with the `sge_qmaster(8)`. Refer to `sge_pe(5)` for details on the PE configuration format. Requires root or manager privileges.
`-mprj` *project*	*Modify project* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Data for the specific project is retrieved (see `project(5)`) and an editor (either `vi(1)` or the editor indicated by `$EDITOR`) is invoked for modifying the project definition. Upon exiting the editor, the modified data is registered. Requires root or manager privileges.
`-mq` *queuename*	*Modify queue configuration* – Retrieves the current configuration for the specified queue, executes an editor (either `vi(1)` or the editor indicated by the `EDITOR` environment variable) and registers the new configuration with the `sge_qmaster(8)`. Refer to `queue_conf(5)` for details on the queue configuration format. Requires root or manager privileges.
`-mqattr` *attr_name val q_name,...* This option is deprecated. Use `-mattr` instead.	*Modify queue attributes* – Allows changing of a single queue configuration attribute in multiple queues with a single command. In all queues contained in the comma separated queue name list the value of the attribute *attr_name* will be overwritten with *val*. All queue attributes can be modified except for *qname* and *qhostname*. Refer to `queue_conf(5)` for details on the queue configuration format. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–msconf`	*Modify scheduler configuration* – The current scheduler configuration (see `sched_conf(5)`) is retrieved, an editor is executed (either `vi(1)` or the editor indicated by `$EDITOR`) and the changed configuration is registered with `sge_qmaster(8)` upon exit of the editor. Requires root or manager privilege. <>
`–mstnode` *node_path=shares,...*	*Modify share tree node* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Modifies the specified share tree node(s) in the share tree (see `share_tree(5)`). The *node_path* is a hierarchical path *([/]node_name[[/.]node_name...]*) specifying the location of an existing node in the share tree. The node is set to the number of specified *shares*. Requires root or manager privileges.
`–mstree`	*Modify share tree* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Modifies the definition of the share tree (see *share_tree(5)*). The present share tree is retrieved and an editor (either *vi(1)* or the editor indicated by $EDITOR) is invoked for modifying the share tree definition. Upon exiting the editor, the modified data is registered with *sge_qmaster(8)*. Requires root or manager privileges.
`–mu` *acl_name*	*Modify user access lists* – Retrieves the current configuration for the specified user access list, executes an editor (either *vi(1)* or the editor indicated by the EDITOR environment variable) and registers the new configuration with the *sge_qmaster(8)*. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–rattr` *obj_spec attr_name val obj_instance,...*	*Replace object attributes* – Allows replacing a single configuration list attribute in multiple instances of an object with a single command. Currently supported objects are the queue and the host configuration being specified as *queue* or *host* in *obj_spec*. The queue *load_thesholds* parameter is an example of a list attribute. With the `-rattr` option, such lists can be replaced, while entries can be added to them with `-aattr`, deleted with `-dattr`, and modified with `-mattr`. The name of the configuration attribute to be modified is specified with *attr_name* followed by *val* defining the new setting of the attribute. The comma separated list of object instances (e.g., the list of queues) to which the changes have to be applied are specified at the end of the command. The following restriction applies: For the *host* object the *load_values* attribute cannot be modified (see *host_conf(5)*). Requires root or manager privileges.
`-Muser` *fname*	*Modify user* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Modify the user defined in fname in the Sun Grid Engine, Enterprise Edition cluster. The format of the user specification is described in `user(5)`. Requires root or manager privileges.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-muser` *user*	*Modify user* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Data for the specific user is retrieved (see *user(5)*) and an editor (either *vi(1)* or the editor indicated by the EDITOR environment variable) is invoked for modifying the user definition. Upon exiting the editor, the modified data is registered. Requires root or manager privileges.
`-sc` *complex_name,...*	*Show complexes* – Display the configuration of one or more complexes.
`-scal` *calendar_name*	*Show calendar* – Display the configuration of the specified calendar.
`-scall`	*Show calendar list* – Show a list of all calendars currently defined.
`-scl`	*Show complex list names* – Show a list of all complexes currently configured.
`-sckpt` *ckpt_name*	*Show checkpoint environment* – Display the configuration of the specified checkpointing environment.
`-sckptl`	*Show checkpoint environment list* – Show a list of the names of all checkpointing environments currently configured.
`-sconf [`*host,...*|*global*`]`	*Show configuration* – Print the cluster configuration being in effect globally or on specified host(s). If the optional comma separated host list argument is omitted or the special string `global` is given, the global cell configuration is displayed. For any other hostname in the list the merger of the global configuration and the host specific configuration is displayed. The format of the host configuration is described in `sge_conf(5)`.
`-sconfl`	*Show configuration list* – Display a list of hosts for which configurations are available. The special host name `global` refers to the cell global configuration.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-se` *hostname*	*Show execution host* – Displays the definition of the specified execution host.
`-secl`	*Show event clients* – Displays the Sun Grid Engine event client list.
`-sel`	*Show execution hosts* – Displays the Sun Grid Engine execution host list.
`-sep`	*Show licensed processors* – Displays a list of number of processors which are licensed per execution host and in total.
`-sh`	*Show administrative hosts* – Displays the Sun Grid Engine administrative host list.
`-sm`	*Show managers* – Displays the managers list.
`-so`	*Show operators* – Displays the operator list.
`-sp` *pe_name*	*Show PE configuration* – Show the definition of the *parallel environment* (PE) specified by the argument.
`-spl`	*Show PE list* – Show a list of all currently defined *parallel environment*s (PEs).
`-sprj` *project*	*Show project* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Shows the definition of the specified project (see `project(5)`).
`-sprjl`	*Show project list* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Shows the list of all currently defined projects.
`-sq` *queue_name[,queue_name,...]*	*Show queues* – Displays one or multiple queues.
`-sql`	*Show queue list* – Show a list of all currently defined queues.

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`-ss`	*Show submit hosts* – Displays the Sun Grid Engine submit host list.
`-ssconf`	*Show scheduler configuration* – Displays the current scheduler configuration in the format explained in `sched_conf(5)`.
`-sstnode` *node_path,...*	*Show share tree node* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Shows the name and shares of the specified share tree node(s) (see `share_tree(5)`). The *node_path* is a hierarchical path *([/]node_name[[/.]node_name...])* specifying the location of a node in the share tree.
`-sstree`	*Show share tree* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Shows the definition of the share tree (see `share_tree(5)`).
`-sss`	*Show scheduler status* – Currently displays the host on which the Sun Grid Engine scheduler is active or an error message if no scheduler is running.
`-su` *acl_name*	*Show user ACL* – Displays a Sun Grid Engine user *access control list* (ACL).
`-sul`	*Show user lists* – Displays a list of names of all currently defined Sun Grid Engine user access control lists (ACLs).

TABLE 4 `qconf` Options *(Continued)*

Option	Description
`–suser` *user,...*	*Show user* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Shows the definition of the specified user(s) (see *user(5)*).
`–suserl`	*Show users* – This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Shows the list of all currently defined users.
`–tsm`	*Trigger scheduler monitoring* – The Sun Grid Engine scheduler `sge_schedd(8)` is forced by this option to print trace messages of its next scheduling run to the file *<sge_root>/<cell>*`/common/schedd_runlog`. The messages indicate the reasons for jobs and queues not being selected in that run. Requires root or manager privileges.

Note – The reasons for job requirements being invalid with respect to resource availability of queues are displayed using the format as described for the `qstat(1)` –F option (see description of Full Format in section Output Formats of the `qstat(1)` manual page.

Environment Variables

TABLE 5 describes the environment variables associated with `qconf`.

TABLE 5 `qconf` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qconf` uses (in the order of precedence): • The name of the cell specified in the environment variable, `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition, specifies the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the TCP port on which `sge_commd(8)` is expected to listen for communication requests. Most installations use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qconf` client resides. By default, the local host is used.

Restrictions

Modifications to a queue configuration do not affect an active queue, taking effect on next invocation of the queue (i.e., the next job).

Files

- Sun Grid Engine master host file – *<sge_root>*`/`*<cell>*`/common/act_qmaster`

See Also

`sge_intro(1)`, `qstat(1)`, `checkpoint(5)`, `complex(5)`, `sge_conf(5)`, `host_conf(5)`, `sge_pe(5)`, `queue_conf(5)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_schedd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qdel(1)

Name

`qdel` – delete Sun Grid Engine jobs from queues.

Syntax

`qdel` [`-f`] [`-help`] [`-verify`] [*job/task_id_list*]

`qdel` [`-f`] [`-help`] [`-verify`] `-u` *user_list* | `-uall`

Description

`Qdel` provides a means for a user, operator, or manager to delete one or more jobs. `Qdel` deletes jobs in the order in which their job identifiers are presented.

Options

TABLE 6 lists options to `qdel`.

TABLE 6 `qdel` Options

Option	Description
`-f`	Force action for running jobs. The job(s) are deleted from the list of jobs registered at `sge_qmaster(8)` even if the `sge_execd(8)` controlling the job(s) does not respond to the delete request sent by `sge_qmaster(8)`. Users which are neither Sun Grid Engine managers nor operators can only use the `-f` option (for their own jobs) if the cluster configuration entry *qmaster_params* contains the flag `ENABLE_FORCED_QDEL` (see `sge_conf(5)`). However, behavior for administrative and non-administrative users differs. Jobs are deleted from the Sun Grid Engine database immediately in case of administrators. Otherwise, a regular deletion is attempted first and a forced cancellation is only executed if the regular deletion was unsuccessful.
`–help`	Prints a listing of all options.
`–u` *username,...* `| -uall`	Deletes only those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the `qdel -uall` command to delete all jobs of all users. If you use the `–u` or `–uall` switch it is be permitted to specify a additional *job/task_id_list*.
`–verify`	Performs no modifications, but just prints what would be done if `–verify` was not present.
`job/`*task_id_list*	Specified by the following form: *job_id*[.*task_range*][,*job_id*[.*task_range*],...] If present, the *task_range* restricts the effect of the `qdel` operation to the job array task range specified as suffix to the job id (see the `–t` option to `qsub(1)` for further details on job arrays). The task range specifier has the form *n[-m[:s]].* The range may be a single number, a simple range of the form *n-m* or a range with a step size. Instead of *job/task_id_list* it is possible to use the keyword, `all`, to modify all jobs of the current user.

Environment Variables

TABLE 7 describes the environment variables associated with `qdel`.

TABLE 7 `qdel` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qdel` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qdel` client resides. Per default the local host is used.

Files

- Sun Grid Engine master host file – <*sge_root*>/<*cell*>/`common/act_qmaster`

See Also

`sge_intro(1)`, `qstat(1)`, `qsub(1)`, `sge_qmaster(8)`, `sge_execd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qhold(1)

Name

`qhold` – hold back Sun Grid Engine jobs from execution.

Syntax

`qhold` [`-h` {`u`|`o`|`s`},...] [`-help`] [*job/task_id_list*]

`qhold` [`-h` {`u`|`o`|`s`},...] [`-help`] `-u` *user_list* | `-uall`

Description

`qhold` provides a means for a user/operator/manager to place so called *holds* on one or more jobs pending to be scheduled for execution. As long as any type of hold is assigned to a job, the job is not eligible for scheduling.

Holds can be removed with the `qrls(1)` or the `qalter(1)` command.

There are three different types of holds:

- **User** – User holds can be assigned and removed by managers, operators and the owner of the jobs.
- **Operator** – Operator holds can be assigned and removed by managers and operators.
- **System** – System holds can be assigned and removed by managers only.

If no hold type is specified with the `-h` option (see below) the user hold is assumed by default.

An alternate way to assign holds to jobs is the `qsub(1)` or the `qalter(1)` command (see the `-h` option).

Options

TABLE 8 lists the options associated with `qhold`.

TABLE 8 `qhold` Options

Option	Description
`-h {u|o|s}`,..	Assign a u(ser), o(perator) or s(system) hold or a combination thereof to one or more jobs.
`-help`	Prints a listing of all options.
`-u` *username,...* | `-uall`	Changes are only made on those jobs which were submitted by users specified in the list of usernames. For managers it is possible to use the `qhold -uall` command to set a hold for all jobs of all users. If you use the `-u` or `-uall` switch it is be permitted to specify a additional *job/task_id_list*.
`job/`*task_id_list*	Specified by the following form: job_id[.task_range][,job_id[.task_range],...] If present, the *task_range* restricts the effect of the *qhold* operation to the job array task range specified as suffix to the job id (see the `-t` option to *qsub(1)* for further details on job arrays). The task range specifier has the form n[-m[:s]]. The range may be a single number, a simple range of the form n-m or a range with a step size. Instead of *job/task_id_list* it is possible to use the keyword 'all' to modify the hold state for all jobs of the current user.

Environment Variables

TABLE 9 describes the environment variables associated with `qhold`.

TABLE 9 `qhold` Environment Variables

Option	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell *qhold* uses (in the order of precedence): • The name of the cell specified in the environment variable SGE_CELL, if it is set. • The name of the default cell, i.e. `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qhold` client resides. Per default the local host is used.

See Also

`sge_intro(1)`, `qalter(1)`, `qrls(1)`, `qsub(1)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qhost(1)

Name

`qhost` – show the status of Sun Grid Engine hosts, queues, jobs.

Syntax

`qhost` [`–F` [*resource_name,...*] [`-help`] [`-h` *host_list*]
[`-j`] [`-l` *resource=val,...*] [`-u` *user,...*]

Description

`qhost` shows the current status of the available Sun Grid Engine hosts, queues and the jobs associated with the queues. Selection options allow you to get information about specific hosts, queues, jobs or users. Without any option `qhost` will display a list of all hosts without queue or job information.

Options

TABLE 10 lists the options associated with `qhost`.

TABLE 10 `qhost` Options

Option	Description
`–F` [*resource_name,...*]	`qhost` will present a detailed listing of the current resource availability per host with respect to all resources (if the option argument is omitted) or with respect to those resources contained in the *resource_name* list. Refer to the description of the "Full Format" in the section, "Output Formats" below for further detail.
`–help`	Prints a listing of all options.
`–h` *host_list*	Prints a list of all hosts contained in *host_list*.

TABLE 10 `qhost` Options *(Continued)*

Option	Description
`-j`	Prints all jobs running on the queues hosted by the shown hosts. This switch calls `-q` implicitly.
`-l` *resource[=value],...*	Defines the resources required by the hosts on which information is requested. Matching is performed on hosts.
`-q`	Show information about the queues hosted by the displayed hosts.
`-u` *user,...*	Display information only on those jobs and queues being associated with the users from the given user list.

Output Formats

Depending on the presence or absence of the `-q` or `-F` and `-j` options, three output formats need to be differentiated.

Default Format

In the default format—that is, without `-q`, `-F`, and `-j` options—following the header line a line is printed for each host consisting of:

- Hostname
- Architecture
- Number of processors
- Load
- Total memory
- Used memory
- Total swapspace
- Used swapspace

Format With Options

If you supply the `-q` option, each host status line also contains extra lines for every queue hosted by the host consisting of:

- Queue name
- Queue type – one of `B` (Batch), `I` (Interactive), `C` (Checkpointing), `P` (Parallel), `T` (Transfer), or combinations thereof
- Number of used and available job slots

- State of the queue – one of `u` (unknown) if the corresponding `sge_execd(8)` cannot be contacted, `a` (alarm), `A` (Alarm), `C` (Calendar suspended), `s` (suspended), `S` (Subordinate), `d` (disabled), `D` (Disabled), `E` (Error), or combinations thereof

If the state is a(alarm) at least one of the load thresholds defined in the *load_thresholds* list of the queue configuration (see `queue_conf(5)`) is currently exceeded, which prevents from scheduling further jobs to that queue.

As opposed to this, the state A(larm) indicates that at least one of the suspend thresholds of the queue (see `queue_conf(5)`) is currently exceeded. This will result in jobs running in that queue being successively suspended until no threshold is violated.

The states s(uspended) and d(isabled) can be assigned to queues and released via the `qmod(1)` command. Suspending a queue will cause all jobs executing in that queue to be suspended.

The states D(isabled) and C(alendar suspended) indicate that the queue has been disabled or suspended automatically via the calendar facility of Sun Grid Engine (see `calendar_conf(5)`), while the S(ubordinate) state indicates, that the queue has been suspend via subordination to another queue (see `queue_conf(5)` for details). When suspending a queue (regardless of the cause) all jobs executing in that queue are suspended too.

If an E(rror) state is displayed for a queue, `sge_execd(8)` on that host was unable to locate the `sge_shepherd(8)` executable on that host in order to start a job. Please check the error logfile of that `sge_execd(8)` for leads on how to resolve the problem. Please enable the queue afterwards via the `-c` option of the `qmod(1)` command manually.

If the `–F` option was used, resource availability information is printed following the host status line. For each resource (as selected in an option argument to `–F` or for all resources if the option argument was omitted) a single line is displayed with the following format:

- A one-letter specifier indicating whether the current resource availability value was dominated by either:
  - `g` – a cluster global
  - `h` – a host total; or
- A second one-letter specifier indicating the source for the current resource availability value, being one of:
  - `l` – a load value reported for the resource,
  - `L` – a load value for the resource after administrator defined load scaling has been applied,
  - `c` – availability derived from the consumable resources facility (see `complexes(5)`)

  - `v` – a default complexes configuration value never overwritten by a load report or a consumable update
  - `f` – a fixed availability definition derived from a non-consumable complex attribute or a fixed resource limit
- After a colon, the name of the resource on which information is displayed
- After an equal sign, the current resource availability value

The displayed availability values and the sources from which they derive are always the minimum values of all possible combinations. Hence, for example, a line of the form “qf:h_vmem=4G” indicates that a queue currently has a maximum availability in virtual memory of 4 Gigabyte, where this value is a fixed value (e.g. a resource limit in the queue configuration) and it is queue dominated, i.e. the host in total may have more virtual memory available than this, but the queue doesn’t allow for more. Contrarily a line “hl:h_vmem=4G” would also indicate an upper bound of 4 Gigabyte virtual memory availability, but the limit would be derived from a load value currently reported for the host. So while the queue might allow for jobs with higher virtual memory requirements, the host on which this particular queue resides currently only has 4 Gigabyte available.

After the queue status line (in case of `-j`) a single line is printed for each job running currently in this queue. Each job status line contains:

- Job ID
- Job name
- Job owner name
- Status of the job – one of t(ransfering), r(unning), R(estarted), s(uspended), S(uspended) or T(hreshold) (see the section, “Reduced Format (Without `-f` and `-F`)” on page 84 for detailed information)
- Start date and time and the function of the job (MASTER or SLAVE - only meaningful in case of a parallel job)
- Priority of the jobs

Environment Variables

TABLE 11 describes the environment variables associated with `qhost`.

TABLE 11 `qhost` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell *qstat* uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qhost` client resides. Per default the local host is used.

Files

- Sun Grid Engine master host file – <*sge_root*>`/`<*cell*>`/common/act_qmaster`

See Also

`sge_intro(1)`, `qalter(1)`, `qconf(1)`, `qhold(1)`, `qmod(1)`, `qstat(1)`, `qsub(1)`, `queue_conf(5)`, `sge_commd(8)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_shepherd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qmake(1)

Name

`qmake` – distributed parallel `make`, scheduling by Sun Grid Engine

Syntax

`qmake` [*options*] `--` [`gmake` *options*]

Description

`qmake` is a parallel, distributed `make(1)`utility. Scheduling of the parallel `make` tasks is done by Sun Grid Engine software. It is based on `gmake` (GNU make), version 3.78.1. Both Sun Grid Engine and `gmake` command line options can be specified. They are separated by `--` characters.

All Sun Grid Engine options valid with `qsub(1)` or `qrsh(1)` can be specified with `qmake`. See `submit(1)` for a description of all Sun Grid Engine command line options. The `make(1)` manual page describes the `gmake` command line syntax.

The syntax of `qmake` makefiles corresponds to `gmake` and is described in the *GNU Make Manual*.

A typical `qmake` call will use the Sun Grid Engine command line options, `-cwd`, to have a scheduled `make` started in the current working directory on the execution host, `-v` *path*, if the Sun Grid Engine environment is not set up in the user's `.cshrc` or `.profile` shell resource file and request slots in a parallel environment (see `sge_pe (5)`).

Examples

The following example, CODE EXAMPLE 1, will request between one and 10 slots in parallel environment `compiling` on the same architecture as the submit host. The `make` tasks will inherit the complete environment of the calling shell. It will be executed as many parallel tasks as slots have been granted by Sun Grid Engine software.

```
qmake -cwd -v PATH -pe compiling 1-10 --
```

CODE EXAMPLE 1

The following example, CODE EXAMPLE 2, will request between one and four slots in parallel environment `make` on the same architecture as the submit host.

```
qmake -cwd -v PATH -- -j 4
```

CODE EXAMPLE 2

The following example, CODE EXAMPLE 3, will request three parallel `make` tasks to be executed on hosts of architecture `solaris`. The submit may be done on a host of any architecture.

```
qmake -cwd -v PATH -l arch=solaris -pe make 3
```

CODE EXAMPLE 3

The following shell script is CODE EXAMPLE 4.

```
#!/bin/sh
qmake -inherit --
```

CODE EXAMPLE 4

You can submit the CODE EXAMPLE 4 shell script by following the pattern of CODE EXAMPLE 5.

```
qsub -cwd -v PATH -pe make 1-10 [further_options] <script>
```

CODE EXAMPLE 5

In CODE EXAMPLE 5, `qmake` will inherit the resources granted for the job submitted under the parallel environment, `make`.

Environment Variables

TABLE 12 describes the environment variables associated with `qmake`.

TABLE 12 `qmake` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell `qmake` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set • The name of the default cell; e.g., `default`
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qmake` client resides. Per default the local host is used.

Known Problems

The following are problems known to be associated with the `qmake` command.

Slow NFS Server

Very low file server performance may lead to problems on depending files.

For example: Host `a` compiles `a.c.` to `a.o.`, host `b` compiles `b.c.` to `b.o.`, and host `c` links program `c` from `a.o.` and `b.o.` In the case of very bad NFS performance, host `c` might not yet see files `a.o.` and `b.o.`

Multiple Commands in One Rule

If multiple commands are executed in one rule, the makefile must ensure that they are handled as *one* command line.

For example, assume that you entered the following.

```
libx.a:
 cd x
 ar ru libx.a x.o
```

In the case above, building `libx.a` will fail if the commands are executed in parallel (and possibly on different hosts). Write the following instead.

```
libx.a:
 cd x ; ar ru libx.a x.o
```

Another effective alternative would be to write the following.

```
libx.a:
 cd x ; \
 ar ru libx.a x.o
```

See Also

`submit(1)`, `sge_pe(5)`, as well as `make(1)` (GNU make manpage) and *The GNU Make Manual* in <*sge_root*>`/3rd_party/qmake`

Copyright

`qmake` contains portions of Gnu Make (`gmake`), which is the copyright of the Free Software Foundation, Inc., Boston, MA, USA, and is protected by the Gnu General Public License.

See `sge_intro(1)` and the information provided in *<sge_root>*`/3rd_party/qmake` for a statement of further rights and permissions.

qmod(1)

Name

`qmod` – modify a Sun Grid Engine queue

Syntax

`qmod [` *options* `] [` *job/task_id_list* `|` *queue_list* `]`

Description

`qmod` enables users classified as *owners* (see `queue_conf(5)` for details) of a workstation to modify the state of Sun Grid Engine queues for his/her machine as well as the state of his/her own jobs. A manager/operator or root can execute `qmod` for any queue and job in a cluster.

Options

TABLE 13 lists the options associated with `qmod`.

TABLE 13 `qmod` Options

Option	Description
`–c`	Clears the error state of the specified queue(s).
`–d`	Disables the queue(s), i.e. no further jobs are dispatched to disabled queues while jobs already executing in these queues are allowed to finish. This option is the successor to the Sun Grid Engine version 3 `-soc` option.
`–e`	Enables the queue(s). This option is the successor to the Sun Grid Engine version 3 `-xsoc` option.
`–f`	Force the modification action for the queue despite the apparent current state of the queue. For example if a queue appears to be suspended but the job execution seems to be continuing the manager/operator can force a suspend operation which will send a `SIGSTOP` to the jobs. In any case, the queue or job status will be set even if the `sge_execd(8)` controlling the queues/jobs cannot be reached. Requires manager/operator privileges.
`–help`	Prints a listing of all options.
`-r`	If applied to queues, reschedules all jobs currently running in this queue. If applied to running jobs, reschedules the jobs.
`–s`	If applied to queues, suspends the queues and any jobs which might be active. If applied to running jobs, suspends the jobs. If a job is both suspended explicitly and via suspension of its queue, a following unsuspend of the queue will not release the suspension state on the job.

TABLE 13 qmod Options *(Continued)*

Option	Description
–us	If applied to queues, unsuspends the queues and any jobs which might be active. If applied to jobs, unsuspends the jobs. If a job is both suspended explicitly and via suspension of its queue, a following unsuspend of the queue will not release the suspension state on the job.
–verify	Performs no modifications, but just prints what would be done if -verify were not present.
job/task_id_list | *queue_list*	The jobs or queues upon which qmod is supposed to operate. The *job/task_id_list* is specified by one of the following forms: • *job_id*[.*task_range*][,*job_id*[.*task_range*],...] • *job_id*[.*task_range*][*job_id*[.*task_range*] ...] If present, the *task_range* restricts the effect of the qmod operation to the job array task range specified as suffix to the *job_id* (see the –t option to qsub(1) for further details on job arrays). The task range specifier has the form: *n*[-*m*[:*s*]][,*n*[-*m*[:*s*]], ...] or: *n*[-*m*[:*s*]][*n*[-*m*[:*s*]] ...] Thus, it consists of a comma- or blank-separated list of range specifiers *n*[-*m*[:*s*]]. The ranges are concatenated to the complete task id range. Each range may be a single number, a simple range of the form *n-m* or a range with a step size.

Environment Variables

TABLE 14 describes the environment variables associated with `qmod`.

TABLE 14 `qmod` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qmod` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition, the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the TCP port on which `sge_commd(8)` is expected to listen for communication requests. Most installations use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qmod` client resides. By default the local host is used.

Files

- Sun Grid Engine master host file – *<sge_root>*`/`*<cell>*`/common/act_qmaster`

See Also

`sge_intro(1)`, `sge_ckpt(1)`, `qstat(1)`, `queue_conf(5)`, `sge_execd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qmon(1)

Name

qmon – X-Windows OSF/Motif graphical user's interface for Sun Grid Engine

Syntax

qmon [*options*]

Description

qmon allows administrators and users to manipulate the Sun Grid Engine system from an X-Window environment. qmon provides various dialogues linked together in multiple ways. For each task the user wishes to accomplish via qmon a corresponding dialogue is provided. There are multiple ways to address the proper dialogue for a certain task:

- The qmon main window that comes up first on startup contains icon buttons for all major administrative and user tasks. A functionality tooltip is displayed when pointing at the different icons.
- A Task pulldown menu button appears in the qmon main window menu bar. Clicking on it opens a list of available tasks. Selecting one of them opens the corresponding dialogue.
- The Task pulldown menu also contains the key accelerators which can be used to invoke the task dialogues directly from the main window by pressing a certain button sequence on the keyboard.
- While navigating through a certain dialogue and its dialogue subhierarchy, links to other dialogues occur whenever a connection between both dialogues is obvious. Pushing the buttons that identify the links opens up the other dialogues.

Options

The supported options are the standard X Toolkit options as described in `X(1)` section, Options. Furthermore, `qmon` supports the options listed in TABLE 15.

TABLE 15 `qmon` Options

Option	Description
`-cmap`	Installs a private color map for `qmon`. This is sometimes useful if other applications have already allocated lots of colors and if `qmon`, therefore, prints corresponding error messages. Using a private color map, however, will result in color map switches whenever you enter or leave `qmon` windows.
`-fontFamily` *{*`big|medium|small`*}*	Notifies `qmon` to use different sized font families for different resolution sizes.
`-help`	Displays usage information.
`-nologo`	Startup without logo.

Dialogues

The following sections describe the dialogues associated with the `qmon` command.

Job Control

The Job Control dialogue provides a folder of tabulated lists of the still pending jobs, already running jobs and recently finished jobs. The dialogue allows for detailed information on the jobs as well as for the deletion and suspension of jobs being selected. In addition the job control dialogue offers links to the Submit dialogue in order to submit new jobs or to change attributes of pending jobs (Qalter button). The shown displayed fields in the tabular display and the jobs displayed can be customized by pressing the Customize button. This customization can be saved to the `~/.qmon_preferences` file and is used on following startups for the initial configuration of the Job Control dialogue.

Queue Control

The Queue Control dialogue with its sub-dialogue hierarchy enables the user to control the status of the Sun Grid Engine queues being actually configured in the system and allows the administrator to add new queues or to modify or delete already existing ones. Each icon button in the top level Queue Control dialogue window represents a configured Sun Grid Engine queue. The icon symbols, the coloring and the text on the buttons informs about the architecture, the status and some basic attributes of the queues. The top level dialogue also allows for deleting those queues previously selected. Queues are selected by clicking with the left mouse button on the icons or into a rectangular area surrounding the buttons.

By pushing the Add or Modify button or using a pop-up menu that is raised when clicking the right mouse button in the icon window of the top level Queue Control dialogue, a sub-dialogue for configuring Sun Grid Engine queues is opened. A queue needs to be selected to use the modify operation. The configuration sub-dialogue allows for definition of the queue and host name or displays the corresponding names in case of a modification. The queue configuration parameters (see `queue_conf(5)`) are subdivided in different categories (General Configuration, Execution Methods, Checkpointing, Load/Suspend Thresholds, Limits, Complexes, User Access, Project Access (only for Sun Grid Engine, Enterprise Edition), Subordinate Queues, Owners) which are selectable by the tab widget area presented in the lower region of the queue configuration sub-dialogue. The administrator may select default values from already configured queues (Clone button). By pushing the Ok button, the definitions are registered with `sge_qmaster(8)`. The Queue Control dialogue can be customized in a similar way as the Job Control dialogue. The settings applied here are also saved in `~/.qmon_preferences`.

Submit

The Job Submission dialogue serves for submitting batch and interactive jobs and is also invoked when changing attributes of pending jobs from the Job Control dialogue explained above (Qalter button). To toggle between batch and interactive jobs please use the Batch/Interactive button at the top of the button column on the right side of the Job Submission screen.

The dialogue consists of a folder containing two job preparation dialogue pages.The most frequently used parameters in the course of a job submission are offered on the General page. A job script has to be defined, all other fields are optional. If the job demands for specification of advanced requirements, the Advanced tab can be used to switch to an enhanced parameter display.

If resource requirements are mandatory for the job, the Request Resources icon button has to be used to pop up the Requested Resources sub-dialogue. This sub-dialogue allows for selection of the required resources of the job and for definition of the quantities in which this resources are to be provided. The Available

Resources are constituted by those complex attributes being declared *requestable* (see `complex(5)` for details). Resource requirements can be made Hard, i.e. they must be met before a job can be started in a queue, or Soft, i.e. they are granted on an as available basis.

Closing the Requested Resources sub-dialogue with the done button books the specified requirement for the job. Pushing the Submit button on the top level Submit dialogue submits the job.

Complex Config

The Complex Config allows the administrator to add new complexes or to modify or delete existing ones (see `complex(5)`). The dialogue offers a selection list for the existing complexes and displays the configuration of the one being selected. By pushing the Delete button, the selected complex is deleted from the configuration. Pushing the Add/Modify button will open a complex configuration dialogue, which allows to create new complexes or which provides the means to change the existing ones. If a new complex is to be created, a name must be defined for it. The name of the complex to be modified is displayed in the same text input filed in case of a modify operation. The complex configuration dialogue provides a tabulated list of the complex entries and an input region for the declaration of new or modified entries. The Add button updates the tabulated list with the new or changed entry and the Ok button registers the additional or modified complex with `sge_qmaster(8)`.

Host Config

Three types of host lists can be maintained via the Host Config dialogue:

- Administration Hosts
- Submit Hosts
- Execution Hosts

The host list to be manipulated is selected via clicking at one of the tabs named correspondingly. The first two host lists only provide for adding or deleting entries, thereby allowing administrative or submit permission for the hosts on the lists, or denying it otherwise respectively. The execution host list entries in addition provide the ability to define scaling factors for the load sensors, consumable complex attributes and access attributes (access, xaccess and projects, xprojects for Sun Grid Engine, Enterprise Edition mode only) as described in `complex(5)`. In a Sun Grid Engine, Enterprise Edition system CPU, memory and I/O usage reported for running jobs can be scaled in addition and the relative performance of a host can be define with the Resource Capability Factor (see `host_conf(5)`).

Cluster Config

This dialogue maintains the cluster global configuration as well as host specific derivatives (see `sge_conf(5)`). When opened, the dialogue displays a selection list for all hosts which have a configuration assigned. The special name "global" refers to the cluster global configuration. By pushing the Add/Modify button a sub-dialogue is opened, which allows for modification of the cluster configuration. For host specific configurations the 'global' host specific configuration fields are set insensitive and only the allowed parameters can be manipulated.

Scheduler Config

The Scheduler Configuration dialogue provides the means to change the behavior of the Sun Grid Engine scheduler daemon `sge_schedd(8)`. The dialogue contains a representation for all scheduler configuration parameters as described in `sched_conf(5)`. It is subdivided in the two sections General Parameters and Load Adjustments which can be selected via the corresponding tabs. The Ok button registers any changes with `sge_qmaster(8)`.

Calendar Config

The Calendar Config allows the administrator to add new calendars or to modify or delete existing ones (see `calendar_conf(5)`). The dialogue offers a selection list for the existing calendars and displays the configuration of the one being selected. By pushing the Delete button, the selected calendar is deleted from the configuration. Pushing the Add/Modify button will open a calendar configuration dialogue, which allows to create new calendars or which provides the means to change the existing ones. The Ok button registers the additional or modified calendar with `sge_qmaster(8)`.

User Config

User permissions are controlled via the User Config dialogue. The tab widget in the left section of the dialogue allows for selecting between

- Configuration of Manager accounts
- Configuration of Operator accounts
- Definition of Usersets
- Definition of User accounts (Sun Grid Engine, Enterprise Edition mode only)

Those user accounts added to the list of manager or operator accounts are given permission to act as managers or operators respectively when accessing Sun Grid Engine under their own account.

The userset lists are used together with the *user_lists* and *xuser_lists* host, queue, project and cluster configuration parameters (see `queue_conf(5)`, `project(5)` and `sge_conf(5)`) to control access of users to hosts, queues, projects (only available in a Sun Grid Engine, Enterprise Edition system) and the entire cluster. A userset is just a collection of user names and UNIX group names. Group names are identified by prefixing them with a "@" sign. The already defined usersets are displayed in a selection list. These lists can be modified and new lists can be created using the Userset definition dialogue.

In a Sun Grid Engine, Enterprise Edition system usersets can be used as Access List (equivalent to their usage in a Sun Grid Engine system) and/or as Department required for the so called Functional Policy and Override Policy (see Ticket Config below).

A Sun Grid Engine, Enterprise Edition system also requires adding accounts having access to the system as entries to the Sun Grid Engine, Enterprise Edition user database (see `user(5).`This can be done with the User sub-dialogue.

The Tickets button in the button list on the right side of the dialogue opens the Ticket Config dialogue (see below). This is also only available in a Sun Grid Engine, Enterprise Edition system.

PE Config

Parallel environment (PE) interfaces can be configured with this dialogue. PE interfaces are necessary to describe the way how parallel programming environments like PVM (Parallel Virtual Machine), MPI (Message Passing Interface) or shared memory parallel systems are to be instantiated and to impose access restrictions onto the PEs. When the dialogue is opened a list of the already configured PEs is displayed together with the current configuration (see `pe_conf(5)`) of the selected PE interface. To add new PE interfaces or to modify existing ones, an Add and a Modify button is available which opens a PE interface configuration sub-dialogue. After applying the changes and quitting this sub-dialogue with the OK button, the new or modified PE interface is registered with `sge_qmaster(8)`.

Checkpoint Config

Checkpointing environment interfaces can be configured with this dialogue. Checkpointing environments are necessary to describe the attributes which the different checkpointing methods and their derivatives on various operating system platforms supported by Sun Grid Engine have. When the dialogue is opened a list of the already configured checkpointing environments is displayed together with the current configuration (see `checkpoint(5)`) of the selected checkpointing environment. To add new checkpointing environment or to modify existing ones, an

Add and a Modify button is available which opens a checkpointing environment configuration sub-dialogue. After applying the changes and quitting this sub-dialogue with the OK button, the new or modified checkpointing environment is registered with `sge_qmaster(8)`.

Ticket Conf

This dialogue offers an overview and editing screen for allocating tickets to the share-based, functional and override scheduling policies. It is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems.

The Deadline Job button opens the User Conf dialogue box. Please change to the Userset sub-dialogue and select the userset named “deadlineusers”. Only users of this userset may submit deadline jobs.

The Share Tree Policy button opens the dialogue for creating and editing the Sun Grid Engine, Enterprise Edition share tree (see `share_tree(5)` and `schedd_conf(5)` for a description of the configuration parameters).

The Functional Policy button opens the dialogue for creating and editing the allocation of the functional shares (see `sched_conf(5)`, `access_list(5)`, `project(5)`, `queue_conf(5)` and `user(5)` for a description of the different types of functional shares and the configurable weighting parameters).

The Override Policy button opens the dialogue for creating and editing the allocation of override tickets (see `access_list(5)`, `project(5)`, `queue_conf(5)` and `user(5)` for a description of the different types of override tickets).

Project Conf

This button opens a dialogue for creating projects. It is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems.

The dialogue offers a selection list for the existing projects and displays the configuration of the one being selected. By pushing the Delete button, the selected project is deleted from the configuration. Pushing the Add/Modify button will open a project configuration dialogue, which allows to create new projects or which provides the means to change the existing ones. Project configuration in essence means giving or denying access to a project for usersets (see User Conf above as well as `project(5)`). The Ok button registers the additional or modified project with `sge_qmaster(8)`.

Browser

The Object Browser dialogue's purpose is manifold: First of all, Sun Grid Engine and `qmon` messages such as notification of error or success concerning a previously taken action can be displayed in the dialogue's output window. Also the standard output and the standard error output of `qmon` can be diverted to the Object Browser output window.

Additionally the Object Browser can be used to display continuous information about Sun Grid Engine objects as the mouse pointer moves over their representation as icons or table entries in other `qmon` dialogues. Currently, only the display of the configuration of two Sun Grid Engine objects in two separate dialogues is supported:

- Queue configurations are displayed as soon as the mouse pointer enters a queue icon in the top level Queue Control dialogue (see above). This facility is activated by pushing the Queue button in the Object Browser dialogue.
- Detailed job information is printed as soon as the user moves the mouse pointer over a line in the Job Control dialogue (see above) being assigned to a running or pending job.
- Additionally job scheduling information is displayed in the browser if the Why ? button in the Job Control dialogue is pressed. In this case the Browser dialogue is opened implicitly and any scheduling related information is displayed.

Exit

The Exit icon button is not linked with a dialogue. Its sole purpose is to close all active `qmon` dialogues and to exit the application.

Resources

The available resources, their meaning and the syntax to be followed in order to modify them are described in the default `qmon` resource file (see the section Files below for the location of the resource file).

Environment Variables

TABLE 16 describes the environment variables associated with `qmon`.

TABLE 16 `qmon` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cel,l `qmon` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set • The name of the default cell; e.g. `default`
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the TCP port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qmon` client resides. Per default the local host is used.

Restrictions

If the line to be entered in an editing window is longer than the width of the window, then the text just runs off the end of the window.

Files

- `qmon` sample resources file – *<sge_root>*`/qmon/Qmon`
- `qmon` system resources file – `/usr/lib/X11/defaults/Qmon`
- `qmon` user resources file – `$HOME/Qmon`
- `qmon` job/queue customization file – `$HOME/.qmon_preferences`

See Also

`sge_intro(1)`, `sge_conf(5)`, `access_list(5)`, `sge_pe(5)`, `calendar_conf(5)`, `complex(5)`, `project(5)`, `queue_conf(5)`, `sched_conf(5)`, `user(5)`, `sge_qmaster(8)`

Copyright

See `sge_intro(1)` and the information provided in *<sge_root>*`/3rd_party/qmon` for a statement of further rights and permissions and for credits to be given to public domain and freeware widget developers.

qrls(1)

Name

`qrls` – release Sun Grid Engine jobs from previous hold states

Syntax

`qrls [ -h` *{*`u|o|s`*},...* `] [ -help ] [` *job/task_id_list* `]`

`qrls [ -h` *{*`u|o|s`*},...* `] [ -help ] -u` *user_list* `| -uall`

Description

`qrls` provides a means for a user, operator, or manager to release so called *holds* from one or more jobs pending to be scheduled for execution. As long as any type of hold is assigned to a job, the job is not eligible for scheduling.

Holds can be assigned to jobs with the `qhold(1)`, `qsub(1)` or the `qalter(1)` commands.

There are three types of holds.

- User – User holds can be assigned and removed by managers, operators and the owner of the jobs.
- Operator – Operator holds can be assigned and removed by managers and operators.
- System – System holds can be assigned and removed by managers only.

If no hold type is specified with the `-h` option (see TABLE 17), the user hold is assumed by default.

An alternate way to release holds is the `qalter(1)` command (see the `-h` option in TABLE 17).

Options

TABLE 17 lists the options associated with `qrls`.

TABLE 17 `qrls` Options

Option	Description
`-h` *{*`u|o|s`*},...*	Releases a u(ser), o(perator) or s(system) hold or a combination thereof from one or more jobs.
`-help`	Prints a listing of all options.
`-u` *username,...* `|` `-uall`	Modifies the hold state of those jobs which were submitted by users specified in the list of user names. For managers it is possible to use the `qrls -uall` command to modify the hold state for jobs of all users. If you use the `-u` or `-uall` switch, you can specify an additional *job/task_id_list*.
job/task_id_list	Specified by the following form: *job_id*`[.`*task_range*`][`*,job_id*`[.`*task_range*`]`*,...*`]` If present, the *task_range* restricts the effect of the operation to the job array task range specified as suffix to the job id (see the `-t` option to `qsub(1)` for further details on job arrays). The task range specifier has the form *n*`[`*-m*`[:`*s*`]]`. The range may be a single number, a simple range of the form *n-m* or a range with a step size. Instead of *job/task_id_list* it is possible to use the keyword `all` to modify all jobs of the current user.

Environment Variables

TABLE 18 describes the environment variables associated with `qrls`.

TABLE 18 `qrls` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qrls` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qrls` client resides. By default, the local host is used.

See Also

`sge_intro(1)`, `qalter(1)`, `qhold(1)`, `qsub(1)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qselect(1)

Name

`qselect` – select queues.

Syntax

`qselect` [`-help`] [`-l` *resource=val,...*] [`-pe` *pe_name,...*]
[`-q` *queue,...*] [`-U` *user,...*]

Description

`qselect` prints a list of Sun Grid Engine queue names corresponding to selection criteria specified in the `qselect` arguments described in TABLE 19. The output of `qselect` can be fed into other Sun Grid Engine commands to apply actions on the selected queue sets. For example, together with the `-mqattr` option to `qconf(1)`, `qselect` can be used to modify queue attributes on a set of queues.

Options

TABLE 19 lists the options associated with `qselect`.

TABLE 19 `qselect` Options

Option	Description
`-help`	Prints a listing of all options.
`-l` *resource*`[`=*value*`]`,...	Defines the resources to be granted by the queues which should be included in the queue list output.

TABLE 19 `qselect` Options *(Continued)*

Option	Description
`-pe` *pe_name,...*	Includes queues into the output which are attached to at least one of the parallel environments enlisted in the comma-separated option argument.
`-q` *queue,...*	Directly specifies the queues to be included in the output. This option usually is only meaningful in conjunction with another *qselect* option to extract a subset of queue names from a list given by `-q`.
`-U` *user,...*	Includes the queues to which the specified users have access in the `qselect` output.

Examples

The following examples demonstrate the uses of the `qselect` command.

```
qselect -l arch=linux
```

CODE EXAMPLE 6

The command in CODE EXAMPLE 6 prints the names of those queues residing on Linux machines.

```
qselect -l arch=linux -U andreas,shannon
```

CODE EXAMPLE 7

In CODE EXAMPLE 7, which builds on CODE EXAMPLE 6, the second command in addition restricts the output to those queues with access permission for the users `andreas` and `shannon`.

```
qconf -mqattr h_vmem=1GB ‘qselect -l arch=linux
```

CODE EXAMPLE 8

CODE EXAMPLE 8 changes the queue attribute `h_vmem` to 1 Gigabyte on queues residing on Linux machines. (See the `qconf(1)` manual page for details on the `-mqattr` option and the `queue_conf(5)` manual page on details of queue configuration entries).

Environment Variables

TABLE 20 describes the environment variables associated with `qselect`.

TABLE 20 `qselect` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qselect` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set • The name of the default cell; e.g., `default`
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition, the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the TCP port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qselect` client resides. By default, the local host is used.

Files

Sun Grid Engine master host file – *<sge_root>*`/`*<cell>*`/common/act_qmaster`

See Also

`sge_intro(1)`, `qconf(1)`, `qmod(1)`, `qstat(1)`, `queue_conf(5)`, `sge_commd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qstat(1)

Name

`qstat` – show the status of Sun Grid Engine jobs and queues

Syntax

`qstat` [`-ext`] [`-f`] [`–F` [*resource_name,...*]] [`-g d`] [`-help`]
[`-j` [*job_list*]] [`-l` *resource=val,...*] [`-ne`]
[`-pe` *pe_name,...*] [`-q` *queue,...*] [`-r`]
[`-s {r|p|s|z|hu|ho|hs|hj|ha|h}[+]]`] [`-t`] [`-U` *user,...*]
[`-u` *user,...*]

Description

`qstat` shows the current status of the available Sun Grid Engine queues and the jobs associated with the queues. Selection options allow you to get information about specific jobs, queues or users. Without any option `qstat` will display only a list of jobs with no queue status information.

Options

TABLE 21 lists the options associated with `qstat`.

TABLE 21 `qstat` Options

Option	Description
`-alarm`	Displays the reason(s) for queue alarm states. Outputs one line per reason containing the resource value and threshold. For details about the resource value, refer to the description of the "Full Format" in the "Output Formats" section below.
`-ext`	This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Displays additional Sun Grid Engine, Enterprise Edition relevant information for each job (see "Output Formats" below).
`-f`	Specifies a "full" format display of information. The `-f` option causes summary information on all queues to be displayed along with the queued job list.
`-F [` *resource_name,...* `]`	Like in the case of `-f`, information is displayed on all jobs as well as queues. In addition, `qstat` will present a detailed listing of the current resource availability per queue with respect to all resources (if the option argument is omitted) or with respect to those resources contained in the *resource_name* list. Refer to the description of the "Full Format" in the "Output Formats" section below.
`-g d`	Displays job arrays verbosely in a one line per job task fashion. By default, job arrays are grouped and all tasks with the same status (for pending tasks only) are displayed in a single line. The job array task id range field in the output (see the "Output Formats" section) specifies the corresponding set of tasks. The `-g` switch currently has only the single option argument `d`. Other option arguments are reserved for future extensions.
`-help`	Prints a listing of all options.

TABLE 21 `qstat` Options *(Continued)*

Option	Description
`-j [`*job_list*`]`	Prints either for all pending jobs or the jobs contained in *job_list* the reason for not being scheduled.
`-l` *resource[=value],...*	Defines the resources required by the jobs or granted by the queues on which information is requested. Matching is performed on queues. The pending jobs are restricted to jobs that might run in one of the above queues.
`-ne`	In combination with `-f` the option suppresses the display of empty queues. This means all queues where actually no jobs are running are not displayed.
`-pe` *pe_name,...*	Displays status information with respect to queues which are attached to at least one of the parallel environments enlisted in the comma separated option argument. Status information for jobs is displayed either for those which execute in one of the selected queues or which are pending and might get scheduled to those queues in principle.
`-q` *queue,...*	Specifies the queue to which job information is to be displayed.
`-r`	Prints extended information about the resource requirements of the displayed jobs. Refer to the “Output Formats” sub-section of the “Expanded Format” section below for detailed information.
`-s {p|r|s|z|hu|ho|hs|hj|ha|h}[+]`	Prints only jobs in the specified state—any combination of states is possible. `-s prs` corresponds to the regular `qstat` output without `-s` at all. To show recently finished jobs, use `-s z`. To display jobs in user/operator/system hold, use the `-s hu/ho/hs` option. The `-s ha` option shows jobs which were submitted with the `qsub -a` command. `qstat -s hj` displays all jobs which are not eligible for execution unless the job has entries in the job dependency list. (See `-a` and `-hold_jid` option to `qsub(1)`).

TABLE 21 `qstat` Options *(Continued)*

Option	Description
`-t`	Prints extended information about the controlled sub-tasks of the displayed parallel jobs. Refer to the “Output Formats” sub-section of the “Expanded Format” section below for detailed information. Sub-tasks of parallel jobs should not be confused with job array tasks (see `-g` option above and `-t` option to `qsub(1)`).
`-U` *user,...*	Displays status information with respect to queues to which the specified users have access. Status information for jobs is displayed either for those which execute in one of the selected queues or which are pending and might get scheduled to those queues in principle.
`-u` *user,...*	Display information only on those jobs and queues being associated with the users from the given user list. Queue status information is displayed if the `-f` or `-F` options are specified additionally and if the user runs jobs in those queues.

Output Formats

Depending on the presence or absence of the `-alarm`, `-f` or `-F` and `-r` and `-t` option three output formats need to be differentiated. PP In case of a Sun Grid Engine, Enterprise Edition system, the `-ext` option may be used to display additional information for each job.

Reduced Format (Without `-f` and `-F`)

Following the header line, a line is printed for each job, consisting of the following.

- Job ID
- Priority of the jobs as assigned to them via the –p option to `qsub(1)` or `qalter(1)` determining the order of the pending jobs list
- Name of the job
- User name of the job owner
- Status of the job – one of `d`(eletion), `t`(ransfering), `r`(unning), `R`(estarted), `s`(uspended), `S`(uspended), `T`(hreshold), `w`(aiting) or `h`(old)

The d(eletion) state indicates hat a qdel(1) has been used to initiate job deletion. The states t(ransfering) and r(unning) indicate that a job is about to be executed or is already executing, whereas the states s(uspended), S(uspended) and T(hreshold) show that an already running job has been suspended. The s(uspended) state is caused by suspending the job via the qmod(1) command, the S(uspended) state indicates that the queue containing the job is suspended and therefore the job is also suspended, and the T(hreshold) state shows that at least one suspend threshold of the corresponding queue was exceeded (see queue_conf(5)) and that the job has been suspended as a consequence. The state R(estarted) indicates that the job was restarted. This can be caused by a job migration or because of one of the reasons described in the -r section of the qsub(1) command.

The states w(aiting) and h(old) only appear for pending jobs. The h(old) state indicates that a job currently is not eligible for execution due to a hold state assigned to it via qhold(1), qalter(1) or the qsub(1) -h option or that the job is waiting for completion of the jobs to which job dependencies have been assigned to the job via the -hold_jid option of qsub(1) or qalter(1).

- Submission or start time and date of the job
- Queue the job is assigned to (for running or suspended jobs only)
- Function of the running jobs (MASTER or SLAVE – the latter for parallel jobs only)
- Job array task id. Will be empty for non-array jobs. See the -t option to qsub(1) and the -g above for additional information

  If the -t option is supplied, each job status line also contains the following.
- Parallel task ID (do not confuse parallel tasks with job array tasks)
- Status of the parallel task – one of r(unning), R(estarted), s(uspended), S(uspended), T(hreshold), w(aiting), h(old), or x(exited)
- CPU, memory, and I/O usage (Sun Grid Engine, Enterprise Edition only)
- Exit status of the parallel task
- Failure code and message for the parallel task

Full Format With -f and -F)

Following the header line a section for each queue separated by a horizontal line is provided. For each queue the information printed consists of the following.

- Queue name
- Queue type – one of B(atch), I(nteractive), C(heckpointing), P(arallel), T(ransfer) or combinations thereof
- Number of used and available job slots
- Load average of the queue host

- Architecture of the queue host
- State of the queue – one of u(nknown) if the corresponding sge_execd(8) cannot be contacted, a(larm), A(larm), C(alendar suspended), s(uspended), S(ubordinate), d(isabled), D(isabled), E(rror) or combinations thereof

If the state is a(larm) at least one of the load thresholds defined in the *load_thresholds* list of the queue configuration (see queue_conf(5)) is currently exceeded, which prevents from scheduling further jobs to that queue.

As opposed to this, the state A(larm) indicates that at least one of the suspend thresholds of the queue (see queue_conf(5)) is currently exceeded. This will result in jobs running in that queue being successively suspended until no threshold is violated.

The states s(uspended) and d(isabled) can be assigned to queues and released via the qmod(1) command. Suspending a queue will cause all jobs executing in that queue to be suspended.

The states D(isabled) and C(alendar suspended) indicate that the queue has been disabled or suspended automatically via the calendar facility of Sun Grid Engine (see calendar_conf(5)), while the S(ubordinate) state indicates, that the queue has been suspend via subordination to another queue (see queue_conf(5) for details). When suspending a queue (regardless of the cause) all jobs executing in that queue are suspended too.

If an E(rror) state is displayed for a queue, sge_execd(8) on that host was unable to locate the sge_shepherd(8) executable on that host in order to start a job. Check the error logfile of that sge_execd(8) for leads on how to resolve the problem. Enable the queue afterwards via the -c option of the qmod(1) command manually.

If the –F option was used, resource availability information is printed following the queue status line. For each resource (as selected in an option argument to –F or for all resources if the option argument was omitted) a single line is displayed with the following format.

- A one-letter specifier indicating whether the current resource availability value was dominated by one of the following:
  - ‘g’ - a cluster global
  - ‘h’ - a host total
  - ‘q’ - a queue related resource consumption
- A second one-letter specifier indicating the source for the current resource availability value, being one of the following:
  - ‘l’ - a load value reported for the resource
  - ‘L’ - a load value for the resource after administrator defined load scaling has been applied

    - ‘c’ - availability derived from the consumable resources facility (see `complexes(5)`), ‘v’ - a default complexes configuration value never overwritten by a load report or a consumable update
    - ‘f’ - a fixed availability definition derived from a non-consumable complex attribute or a fixed resource limit
- After a colon, the name of the resource on which information is displayed
- After an equal sign, the current resource availability value

The displayed availability values and the sources from which they derive are always the minimum values of all possible combinations. Hence, for example, a line of the form “qf:h_vmem=4G” indicates that a queue currently has a maximum availability in virtual memory of 4 Gigabyte, where this value is a fixed value (e.g. a resource limit in the queue configuration) and it is queue dominated, i.e. the host in total may have more virtual memory available than this, but the queue doesn’t allow for more. Contrarily a line “hl:h_vmem=4G” would also indicate an upper bound of 4 Gigabyte virtual memory availability, but the limit would be derived from a load value currently reported for the host. So while the queue might allow for jobs with higher virtual memory requirements, the host on which this particular queue resides currently only has 4 Gigabyte available.

If the `–alarm` option was used, information about resources is displayed, that violate load or suspend thresholds.

The same format as with the `-F` option is used with the following extensions.

- The line starts with the keyword, `alarm`.
- Appended to the resource value is the type and value of the appropriate threshold.

After the queue status line (in case of `–f`) or the resource availability information (in case of `–F`) a single line is printed for each job running currently in this queue. Each job status line contains

- Job ID
- Job name
- Job owner name
- Status of the job – one of t(ransfering), r(unning), R(estarted), s(uspended), S(uspended) or T(hreshold) (see the Reduced Format section for detailed information)
- Start date and time and the function of the job (master or slave, which is only meaningful in the case of a parallel job)
- Priority of the jobs

If the `–t` option is supplied, each job status line also contains the following.

- Task ID

- Status of the task – one of r(unning), R(estarted), s(uspended), S(uspended), T(hreshold), w(aiting), h(old), or x(exited) (see the Reduced Format section for detailed information)
- CPU, memory, and I/O usage (Sun Grid Engine, Enterprise Edition only)
- Exit status of the task
- Failure code and message for the task

Following the list of queue sections a *pending jobs* list may be printed in case jobs are waiting for being assigned to a queue. A status line for each waiting job is displayed being similar to the one for the running jobs. The differences are that the status for the jobs is w(aiting) or h(old), that the submit time and date is shown instead of the start time and that no function is displayed for the jobs.

In very rare cases, e.g. if `sge_qmaster(8)` starts up from an inconsistent state in the job or queue spool files or if the clean queue (`-cq`) option of `qconf(1)` is used, `qstat` cannot assign jobs to either the running or pending jobs section of the output. In this case as job status inconsistency (e.g. a job has a running status but is not assigned to a queue) has been detected. Such jobs are printed in an *error jobs* section at the very end of the output. The `ERROR JOBS` section should disappear upon restart of `sge_qmaster(8)`. Please contact your Sun Grid Engine support representative if you feel uncertain about the cause or effects of such jobs.

Expanded Format (With `-r`)

If the `-r` option was specified together with `qstat`, the following information for each displayed job is printed (a single line for each of the following job characteristics).

- Hard and soft resource requirements of the job as specified with the `qsub(1) -l` option
- Requested parallel environment including the desired queue slot range (see `-pe` option of `qsub(1)`)
- Requested checkpointing environment of the job (see the `qsub(1) -ckpt` option)
- In case of running jobs, the granted parallel environment with the granted number of queue slots

Enhanced Sun Grid Engine, Enterprise Edition Output (With `-ext`)

For each job the following additional items are displayed.

- `project` – The project to which the job is assigned as specified in the `qsub(1) -P` option.

- `department` – The department, to which the user belongs (use the `-sul` and `-su` options of `qconf(1)` to display the current department definitions).
- `deadline` – The deadline initiation time of the job as specified with the `qsub(1) -dl` option
- `cpu` – The current accumulated CPU usage of the job
- `mem` – The current accumulated memory usage of the job
- `io` – The current accumulated IO usage of the job
- `tckts` – The total number of tickets assigned to the job currently
- `ovrts` – The override tickets as assigned by the `-ot` option of `qalter(1)`
- `otckt` – The override portion of the total number of tickets assigned to the job currently
- `dtckt` – The deadline portion of the total number of tickets assigned to the job currently
- `ftckt` – The functional portion of the total number of tickets assigned to the job currently
- `stckt` – The share portion of the total number of tickets assigned to the job currently
- `share` – The share of the total system to which the job is entitled currently

Environment Variables

TABLE 22 describes the environment variables associated with `qstat`.

TABLE 22 `qstat` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qstat` uses (in the order of precedence): • The name of the cell specified in the environment variable, `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.

TABLE 22 `qstat` Environment Variables *(Continued)*

Name of Variable	Description
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the TCP port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qstat` client resides. Per default the local host is used.

Files

Sun Grid Engine master host file – *<sge_root>*`/`*<cell>*`/common/act_qmaster`

See Also

`sge_intro(1)`, `qalter(1)`, `qconf(1)`, `qhold(1)`, `qhost(1)`, `qmod(1)`, `qsub(1)`, `queue_conf(5)`, `sge_commd(8)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_shepherd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qtcsh(1)

Name

`qtcsh` – `tcsh` v6.09 with transparent remote execution by use of `qrsh`.

Syntax

`qtcsh [ tcsh` *options* `| -ABLR ]`

Description

`qtcsh` is an extension to the popular `csh(1)` derivative, `tcsh`. It allows the transparent remote execution commands entered in `qtcsh` controlled via Sun Grid Engine. `qtcsh` can be used as interactive command interpreter as well as for the processing of `tcsh` shell scripts.

When invoked, `qtcsh` identifies which commands are to be run remotely and which are not. For this purpose the files *<sge_root>*/*<cell>*`/common/qtask` and `~/.qtask` are processed. Each line in these files defines a command that is intended to be run remotely (see `qtask(5)` for a definition of the file format). The `.qtask` file in the user's home direcory contains the user's remote task specification, while the file in the common directory is maintained by the administrator and defines a cluster-wide default behavior. The contents of the administrator supplied `qtask(5)` file are completely overridden in case there is an appropriate entry in the users `qtask(5)` file. This is prevented in case an exclamation mark is prefixed to the command name in the administrators qtask file.

`qtcsh` always attempts to start the designated tasks remotely via `qrsh(1)`. Exceptions are the following.

- If the user enters such commands via a relative or absolute pathname instead of the standalone command name. See `qtask(5)` for more information.
- If the environment variable `JOB_ID` is set and thus `qtcsh` assumes that execution already happens remotely within a Sun Grid Engine job and thus executes tasks locally. This avoids unwanted recursions but can be overridden by the command-line option `–R` and the built-in command `qrshmode –R` (see corresponding descriptions below).
- if `qtcsh` cannot establish a connection of Sun Grid Engine during startup. This allows to use `qtcsh` as login shell without the danger of being blocked when no Sun Grid Engine service is available.

`qtcsh` can operate in three different modes, determining whether:

- Tasks are executed remotely
- Immediate or batch execution is requested
- Status output is verbose or only in case of any errors

These modes either can be controlled by the command-line switches described below during `qtcsh` invocation or within an executing `qtcsh` via the built-in command, `qrshmode`, as described in the section "Built-In Commands" on page 92.

Options

The options listed in TABLE 23 are special to `qtcsh`. Refer to the `tcsh(1)` documentation for the explanation of further options.

TABLE 23 `qtcsh` Options

Option	Description
`–A`	Switches `qtcsh` in verbose mode causing diagnostic output in case of remote execution.
`–B`	Switches remote task execution to batch mode. Tasks submitted to Sun Grid Engine will be queued if they cannot start immediately. As a consequence, `qtcsh` may block until the queued task can be started by Sun Grid Engine. While this behavior probably is undesirable during an interactive session, it may be very useful for execution of shell scripts through `qtcsh` as it avoids failure of the scripts due to temporarily unavailable resources for particular tasks.
`–L`	Switches off default behavior of remote execution of commands. Causes all commands to be executed locally even if they are contained in one of the `qtask(5)` files.
`–R`	Enforces remote execution of commands even if `JOB_ID` is set as environment variable.

Built-In Commands

This section describes only the additional shell built-in commands that are not available in standard `tcsh(1)`.

`qrshmode [–ANBILR]`

Without options, the current operational mode of `qtcsh` is displayed. The options have the following effect.

- `-A` – Switch to verbose output mode.
- `-N` – Switch to non-verbose output mode.
- `-B` – Switch to batch execution mode.
- `-I` – Switch to immediate execution mode.
- `-L` – Always execute commands locally.
- `-R` – Execute configured commands remotely.

Environment Variables

TABLE 24 lists the environment variables associated with `qtsch`.

TABLE 24 `qtsch` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `qtcsh` uses (in the order of precedence): • The name of the cell specified in the environment variable, `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the TCP port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `qtcsh` client resides. By default, the local host is used.

Files

- User `qtask` file – `~/.qtask`
- Cluster `qtask` file – <*sge_root*>/<*cell*>`/common/qtask`

See Also

`sge_intro(1)`, `qrsh(1)`, `qtask(5)`, as well as `tcsh(1)` in <*sge_root*>`/3rd_party/qtcsh`

Copyright

`qtcsh` contains portions of `tcsh` which is copyrighted by The Regents of the University of California. Therefore, the following note applies: This product includes software developed by the University of California, Berkeley and its contributors.

See `sge_intro(1)` and the information provided in *<sge_root>*`/3rd_party/qtcsh` for a statement of further rights and permissions.

submit(1)

Name

`qsub` – Submit a batch job to Sun Grid Engine.

`qsh` – Submit an interactive X-windows session to Sun Grid Engine.

`qlogin` – Submit an interactive login session to Sun Grid Engine.

`qrsh` – Submit an interactive `rsh` session to Sun Grid Engine.

`qalter` – Modify a pending batch job of Sun Grid Engine.

`qresub` – Submit a copy of an existing Sun Grid Engine job.

Syntax

`qsub [` *options* `] [` *scriptfile* `| -- [` *script_args* `]]`

`qsh [` *options* `] [ --` *xterm_args* `]`

`qlogin [` *options* `]`

`qrsh [` *options* `] [` *command* `[` *command_args* `]]`

`qalter [` *options* `]` *job/task_id_list* `[ -- [` *script_args* `]]`

`qalter [` *options* `] -u` *user_list* `| -uall [ -- [` *script_args* `]]`

`qresub [` *options* `]` *job_id_list*

Description

`qsub` submits batch jobs to the Sun Grid Engine queuing system. Sun Grid Engine supports single and multiple node jobs. *scriptfile* contains the commands to be run by the job using a shell (for example, `sh(1)` or `csh(1)`). Arguments to the job script are given by *script_args*. Sun Grid Engine flags may be entered as arguments to qsub or as embedded flags in the *scriptfile* if the first two characters of a script line either match '#$' or are equal to the prefix string defined with the -C option described below.

qsh submits an interactive X-windows session to Sun Grid Engine. An xterm(1) is brought up from the executing machine with the display directed either to the X-server indicated by the DISPLAY environment variable or as specified with the –display qsh option. Interactive jobs are not spooled if no resource is available to execute them. They are either dispatched to a suitable machine for execution immediately or the user submitting the job is notified by qsh that appropriate resources to execute the job are not available. xterm_args are passed to the xterm(1) executable.

qlogin is similar to qsh in that it submits an interactive job to the queueing system. It does not open an xterm(1) window on the X display, but uses the current terminal for user I/O. Usually, qlogin establishes a telnet(1) connection with the remote host, using standard client- and server-side commands. These commands can be configured with the qlogin_daemon (server-side, Sun Grid Engine telnetd if not set, otherwise something like /usr/sbin/in.telnetd) and qlogin_command (client-side, Sun Grid Engine telnet if not set, otherwise something like /usr/bin/telnet) parameters in the global and local configuration settings of sge_conf(5). The client side command is automatically parameterized with the remote host name and port number to connect to (i.e. resulting in an invocation like /usr/bin/telnet my_exec_host 2442). qlogin is invoked exactly like qsh and its jobs can only run on INTERACTIVE queues. qlogin jobs can only be used if the sge_execd(8) is running under the root account.

qrsh is similar to qlogin in that it submits an interactive job to the queuing system. It uses the current terminal for user I/O. Usually, qrsh establishes a rsh(1) connection with the remote host. If no command is given to qrsh, a rlogin(1) session is established. The server-side commands used can be configured with the rsh_daemon and rlogin_daemon parameters in the global and local configuration settings of sge_conf(5). A Sun Grid Engine rshd or rlogind is used, if the parameters are not set or otherwise something like /usr/sbin/in.rshd or /usr/sbin/in.rlogind needs to be configured. On the client-side, the rsh_command and rlogin_command parameters can be set in the global and local configuration settings of sge_conf(5). If they are not set, rsh(1) and rlogin(1) binaries delivered with Sun Grid Engine are used. Use the cluster configuration to integrate mechanisms like ssh or the rsh(1) and rlogin(1) facilities supplied with the operating system.

qrsh jobs can only run in INTERACTIVE queues unless the option -now no is used (see below). They can only be used, if the sge_execd(8) is running under the root account.

qrsh provides an additional feature useful for the integration with interactive tools providing a specific command shell. If the environment variable QRSH_WRAPPER is set when qrsh is invoked, the command interpreter pointed to by QRSH_WRAPPER will be executed to run qrsh commands instead of the user's login shell or any shell specified in the qrsh command line.

qalter can be used to change the attributes of pending jobs. Once a job is executing, changes are no longer possible. For job arrays, for which a part of the tasks can be pending and another part can be running (see the -t option below), modifications with qalter only affect the pending tasks. qalter can change most of the characteristics of a job (see the corresponding statements in the Options section below), including those which were defined as embedded flags in the script file (see above).

qresub allows you to create jobs as copies from existing pending or running jobs. The copied jobs will have exactly the same attributes as the ones from which they are copied, but a new job ID. The only modification to the copied jobs supported by qresub is to assign a hold state with the -h option. This can be used to first copy a job and then change its attributes via qalter.

For qsub, qsh, qrsh, and qlogin the administrator and the user may define default request files (see sge_request(5)) which can contain any of the options described below. If an option in a default request file is understood by qsub and qlogin but not by qsh the option is silently ignored if qsh is invoked. Thus you can maintain shared default request files for both qsub and qsh.

A cluster-wide default request file may be placed under *$sge_root/$sge_cell*/common/sge_request. User private default request files are processed under the locations $HOME/.sge_request and $cwd/.sge_request. The working directory local default request file has the highest precedence, then the home directory located file and then the cluster global file. The option arguments, the embedded script flags and the options in the default request files are processed in the following order.

1. Left to right in the script line

2. Left to right in the default request files

3. Top to bottom of the script file (qsub only)

4. Top to bottom of default request files

5. Left to right of the command line

In other words, the command line can be used to override the embedded flags and the default request settings. The embedded flags, however, will override the default settings.

Note – The `-clear` option can be used to discard any previous settings at any time in a default request file, in the embedded script flags, or in a command-line option. It is, however, not available with `qalter`.

The options described in TABLE 25 can be requested either hard or soft. By default, all requests are considered hard until the `–soft` option (see below) is encountered. The hard/soft status remains in effect until its counterpart is encountered again. If all the hard requests for a job cannot be met, the job will not be scheduled. Jobs which cannot be run at the present time remain spooled.

Options

TABLE 25 lists and describes the options associated with the various submit commands.

TABLE 25 `submit` Command Options

Option	Description
`-@` *optionfile*	Forces `qsub`, `qrsh`, `qsh`, or `qlogin` to use the options contained in *optionfile*. The indicated file may contain all valid options. Comment lines are starting with a # sign.
`-a` *date_time*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Defines or redefines the time and date at which a job is eligible for execution. *Date_time* conforms to *[[CC]]YY]MMDDhhmm.[ss]*, where: *CC* denotes the century in 2 digits. *YY* denotes the year in 2 digits. *MM* denotes the month in 2 digits. *DD* denotes the day in 2 digits. *hh* denotes the hour in 2 digits. mm denotes the minute in 2 digits. ss denotes the seconds in 2 digits (default 00). If any of the optional date fields is omitted, the corresponding value of the current date is assumed. Usage of this option may cause unexpected results if the clocks of the hosts in the Sun Grid Engine pool are out of sync. Also, the proper behavior of this option very much depends on the correct setting of the appropriate timezone, e.g. in the TZ environment variable (see `date(1)` for details), when the Sun Grid Engine daemons `sge_qmaster(8)` and `sge_execd(8)` are invoked. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however
`-ac` *variable[=value],...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin`, and `qalter` only. Adds the given name/value pair(s) to the job's context. *Value* may be omitted. Sun Grid Engine appends the given argument to the list of context variables for the job. Multiple `-ac`, `-dc`, and `-sc` options may be given. The order is important here. `qalter` allows changing this option even while the job executes.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–A` *account_string*	Available for `qsub`, `qrsh`, `qsh`, `qlogin`, and `qalter` only. Identifies the account to which the resource consumption of the job should be charged. The *account_string* may be any arbitrary ASCII alphanumeric string but may contain no blank or separator characters. The underbar '_' is considered a non-separator. In the absence of this parameter Sun Grid Engine will place the default account string `sge` in the accounting record of the job. `qalter` allows changing this option even while the job executes.
`–c` *occasion_specifier*	Available for `qsub` and `qalter` only. Defines or redefines whether the job should be checkpointed, and if so, under what circumstances. The specification of the checkpointing occasions with this option overwrites the definitions of the *when* parameter in the checkpointing environment (see `checkpoint(5)`) referenced by the `qsub –ckpt` switch. Possible values for *occasion_specifier* are `n` – No checkpoint is performed. `s` – Checkpoint when batch server is shut down. `m` – Checkpoint at minimum CPU interval. `x` – Checkpoint when job gets suspended. *<interval>* – Checkpoint in the specified time interval. The minimum CPU interval is defined in the queue configuration (see `queue_conf(5)` for details). `<interval>` has to be specified in the format `hh:mm:ss`. The maximum of `<interval>` and the queue's minimum CPU interval is used if `<interval>` is specified. This is done to ensure that a machine is not overloaded by checkpoints being generated too frequently.
`–ckpt` *ckpt_name*	Available for `qsub` and `qalter` only. Selects the checkpointing environment (see `checkpoint(5)`) to be used for a checkpointing the job. Also declares the job to be a checkpointing job.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–clear`	Available for `qsub`, `qrsh`, `qsh`, and `qlogin` only. Causes all elements of the job to be reset to the initial default status prior to applying any modifications (if any) appearing in this specific command.
`–cwd`	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Execute the job from the current working directory. This switch will activate Sun Grid Engine's path aliasing facility, if the corresponding configuration files are present (see `sge_aliases(5)`). In case of `qalter`, the previous definition of the current working directory will be overwritten, if `qalter` is executed from a different directory than the preceding `qsub` or `qalter`. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
`–C` *prefix_string*	Available for `qsub` only. *prefix_string* defines the prefix that declares a directive to `qsub` in the job's script file. The prefix is not a job attribute, but affects the behavior of `qsub`. If the `-C` option is presented with the value of the directive prefix as a null string, `qsub` will not scan the script file. The directive prefix consists of two ASCII characters which when appearing in the first two bytes of a script line indicate that what follows is a Sun Grid Engine command (default is `#$`). The user should be aware that changing the first delimiter character can produce unforeseen side effects. If the script file contains anything other than a # character in the first byte position of the line, the shell processor for the job will reject the line and may exit the job prematurely. If the `-C` option is present in the script file, it is ignored.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-dc` *variable,...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Removes the given variable(s) from the job's context. Multiple `-ac`, `-dc`, and `-sc` options may be given. The order is important here. `qalter` allows changing this option even while the job executes.
`-display` *display_specifier*	Available for `qsh` only. Directs `xterm(1)` to use *display_specifier* in order to contact the X server.
`-dl` *date_time*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Specifies the deadline initiation time in *[[CC]YY]DDhhmm[.SS]* format (see `-a` option above). The deadline initiation time is the time at which a deadline job has to reach top priority to be able to complete within a given deadline. Before the deadline initiation time the priority of a deadline job will be raised steadily until it reaches the maximum as configured by the Sun Grid Engine administrator. This option is applicable for users allowed to submit deadline jobs only.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-e` *[hostname:]path,...*	Available for `qsub` and `qalter` only. Defines or redefines the path used for the standard error stream of the job. If the *path* constitutes an absolute path name, the error-path attribute of the job is set to its value including the *hostname*. If the path name is relative, Sun Grid Engine expands *path* either with the current working directory path in case the `-cwd` (see above) switch is also specified or with the home directory path otherwise. If *hostname* is present, the standard error stream will be placed under the corresponding location if the job runs on the specified host. By default the file name for standard error has the form *job_name*.e*job_id* and *job_name*.e*job_id*.*task_id* for job array tasks (see the `-t` option below). If *path* is a directory, the standard error stream of the job will be put in this directory under the default file name. If the pathname contains certain pseudo environment variables, their value will be expanded at runtime of the job and will be used to constitute the standard error stream path name. The following pseudo environment variables are supported currently: *$HOME* – home directory on execution machine *$USER* – user ID of job owner *$JOB_ID* – current job ID *$JOB_NAME* – current job name (see `-N` option) *$HOSTNAME* – name of the execution host *$TASK_ID* – job array task index number Alternatively to *$HOME* the tilde sign "~" can be used as common in `csh(1)` or `ksh(1)`. The "~" sign also works in combination with user names, so that "~<user>" expands to the home directory of <user>. Using another user ID than that of the job owner requires corresponding permissions, of course. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–hard`	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Signifies that all resource requirements following in the command line will be hard requirements and must be satisfied in full before a job can be scheduled. As Sun Grid Engine scans the command line and script file for Sun Grid Engine options and parameters it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the `–soft` option (see below) is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice-to-have, but not essential". If the `–hard` flag is encountered at a later stage of the scan, all resource requests following it once again become "essential". The `–hard` and `–soft` options in effect act as "toggles" during the scan.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-h | -h {u|s|o|n|U|O|S}`...	Available for `qsub`, `qrsh`, `qsh`, `qlogin`, `qalter` and `qresub`. List of holds to place on the job. `u` – Denotes a user hold. `s` – Denotes a system hold. `o` – Denotes a operator hold. `n` – Denotes no hold. As long as any hold other than 'n' is assigned to the job the job is not eligible for execution. Holds can be released via `qalter` and `qrls(1)`. In case of `qalter` this is supported by the following additional option specifiers for the `-h` switch: `U` – Removes a user hold. `S` – Removes a system hold. `O` – Removes a operator hold. Sun Grid Engine managers can assign and remove all hold types, Sun Grid Engine operators can assign and remove user and operator holds and users can only assign or remove user holds. In the case of `qsub`, only user holds can be placed on a job and thus only the first form of the option with the `-h` switch alone is allowed. As opposed to this, `qalter` requires the second form described above. An alternate means to assign hold is provided by the `qhold(1)` facility. If the job is a job array (see the `-t` option below), all tasks specified via `-t` are affected by the `-h` operation simultaneously. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
`-help`	Prints a listing of all options.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-hold_jid` *[job_id | job_name],...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Defines or redefines the job dependency list of the submitted job. A reference by job name is only accepted if the referenced job is owend by the same user as the refering job. The submitted job is not eligible for execution unless all jobs referenced in the coma-separated *job_id* and/or *job_name* list have completed successfully. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
`–inherit`	Available only for `qrsh` and `qmake(1)`. `qrsh` allows you to start a task in an already scheduled parallel job. The option `–inherit` tells `qrsh` to read a job id from the environment variable *JOB_ID* and start the specified command as a task in this job. Please note that in this case, the hostname of the host where the command shall be executed, must precede the command to execute; the syntax changes to `qrsh-inherit` [*other options*] *hostname command* [*command_args*] Note also, that in combination with `–inherit`, most other command line options will be ignored. Only the options `–verbose`, `–v` and `–V` will be interpreted. As a replacement to option `–cwd`, use `–v PWD`. Usually a task should have the same environment (including the current working directory) as the corresponding job, so specifying the option `–V` should be suitable for most applications. If in your system the commd port is not configured as service, but via environment variable `COMMD_PORT`, make sure that this variable is set in the enviroment when calling `qrsh` or `qmake` with option `–inherit`. If you call `qrsh` or `qmake` with option `–inherit` from within a job script, export `COMMD_PORT` with the `submit` option or special comment: `-v COMMD_PORT`

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-j y|n`	Available for `qsub` and `qalter` only. Specifies whether or not the standard error stream of the job is merged into the standard output stream. If both the `-j y` and the `-e` options are present, Sun Grid Engine sets, but ignores the error-path attribute. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
`-l` *resource=value,...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Launch the job in a Sun Grid Engine queue meeting the given resource request list. In case of `qalter` the previous definition is replaced by the specified one. `complex(5)` describes how a list of available resources and their associated valid value specifiers can be obtained. There may be multiple `-l` switches in a single command. You may request multiple `-l` options to be soft or hard both in the same command line. In case of a serial job multiple `-l` switches refine the definition for the sought queue. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–m b|e|a|s|n`,...	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the `–M` option described below. The option arguments have the following meaning: `b` – Mail is sent at the beginning of the job. `e` – Mail is sent at the end of the job. `a` – Mail is sent when the job is aborted or rescheduled. `s` – Mail is sent when the job is suspended. `n` – No mail is sent. Currently, no mail is sent when a job is suspended. For `qsh` and `qlogin` mail at the beginning or end of the job is suppressed when it is encountered in a default request file. `qalter` allows changing the `b`, `e`, and `a` option arguments even while the job executes. The modification of the `b` option argument will only be in effect after a restart or migration of the job, however.
`–M` *user[@host],...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Defines or redefines the list of users to which the server that executes the job has to send mail, if the server sends mail about the job. Default is the job owner at the originating host. `qalter` allows changing this option even while the job executes.

TABLE 25 submit Command Options *(Continued)*

Option	Description
–masterq *queue,...*	Available for qsub, qsh, qrsh, qlogin and qalter. Only meaningful for parallel jobs, i.e. together with the -pe option. Defines or redefines a list of queues which may be used to become the so called *master queue* of this parallel job. The *master queue* is defined as the queue where the parallel job is started. The other queues to which the parallel job spawns tasks are called *slave queues*. A parallel job only has one *master queue*. This parameter has all the properties of a resource request and will be merged with requirements derived from the –l option described above. qalter allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
–notify	Available for qsub, qrsh, qsh, qlogin and qalter only. This flag, when set, causes Sun Grid Engine to send "warning" signals to a running job prior to sending the signals themselves. If a SIGSTOP is pending the job will receive a SIGUSR1 several seconds before the SIGSTOP. If a SIGKILL is pending the job will receive a SIGUSR2 several seconds before the SIGKILL. The amount of time delay is controlled by the notify parameter in each queue configuration (see queue_conf(5)). The Linux operating system "misuses" the user signals SIGUSR1 and SIGUSR2 in its current Posix thread implementation. You might not want to use the –notify option if you are running threaded applications in your jobs under Linux. qalter allows changing this option even while the job executes.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-now y[`*es*`]|n[`*o*`]`	Available for `qsub`, `qrsh`, `qsh`, and `qlogin`. `-now y` tries to start the job immediately or not at all. The command returns 0 on success, or 1 on failure (also if the job could not be scheduled immediately). `-now y` is default for `qsh`, `qlogin` and `qrsh.` With the `-now n` option, the job will be put into the pending queue, if it cannot be executed immediately. `-now n` is default for `qsub`.
`-N` *name*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. The name of the job. The name can be any printable set of characters, starting with an alphabetic character. If the `-N` option is not present, Sun Grid Engine assigns the name of the job script to the job after any directory path name has been removed from the script name. If the script is read from standard input, the job name defaults to `STDIN`. In case of `qsh` or `qlogin`—and if the `-N` option is absent—the string, `INTERACT`, is assigned to the job. `qalter` allows changing this option even while the job executes.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description

`-noshell`	Available only for `qrsh` with a command line. Do not start the command line given to qrsh in a user's login shell, but execute it without the wrapping shell.
	The option can be used to speed up execution as some overhead like the shell startup and sourcing the shell resource files is avoided.
	The option can only be used, if no shell specific command line parsing is required. If the command line contains shell syntax like environment variable substitution or (back) quoting, a shell must be started. In this case, either do not use the -noshell option or include the shell call in the command line.
	Example:
	`qrsh echo '$HOSTNAME'`
	Alternative call with the `-noshell` option:
	`qrsh -noshell /bin/tcsh -f -c 'echo $HOSTNAME'`

`–nostdin`	Available only for `qrsh`. Suppress the input stream `STDIN` - `qrsh` will pass the option `-n` to the `rsh(1)` command. This is especially useful, if multiple tasks are executed in parallel using `qrsh`; e.g., in a `make(1)` process. Which process would get the input would be undefined.
`–o` *[hostname:]path,...*	Available for `qsub` and `qalter` only. The path used for the standard output stream of the job. The *path* is handled as described in the `–e` option for the standard error stream. By default the file name for standard output has the form *job_name*.o*job_id* and *job_name*.o*job_id*.*task_id* for job array tasks (see `–t` option below). `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-ot` *override_tickets*	Available for `qalter` only. This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Changes the number of override tickets for the specified job. Requires manager/operator privileges.
`-P` *project_name*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. This option is only supported in case of a Sun Grid Engine, Enterprise Edition system. It is not available for Sun Grid Engine systems. Specifies the project to which this job is assigned. The administrator needs to give permission to individual users to submit jobs to a specific project. (See `-aprj` option to `qconf(1)`).

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–p` *priority*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range -1023 to 1024. The default priority value for the jobs is `0`. In a Sun Grid Engine system, users may only decrease the priority of their jobs. Sun Grid Engine managers and administrators may also increase the priority associated with jobs. If a pending job has higher priority, it is earlier eligible for being dispatched by the Sun Grid Engine scheduler. The job priority has no effect on running jobs in Sun Grid Engine. In Sun Grid Engine, Enterprise Edition, the job priority influences the Share Tree Policy and the Functional Policy. It has no effect on the Deadline and Override Policies (see `share_tree(5)`, `sched_conf(5)` and the *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide* for further information on the resource management policies supported by Sun Grid Engine, Enterprise Edition). In case of the Share Tree Policy, users can distribute the tickets, to which they are currently entitled, among their jobs using different priorities assigned via `–p`. If all jobs have the same priority value, the tickets are distributed evenly. Jobs receive tickets relative to the different priorities otherwise. Priorities are treated like an additional level in the share tree in the latter case. In connection with the Functional Policy, the priority can be used to weight jobs within the functional job category. Again tickets are distributed relative to any uneven priority distribution treated as a virtual share distribution level underneath the functional job category. If both, the Share Tree and the Functional Policy are active, the job priorities will have an effect in both policies and the tickets independently derived in each of them are added up to the total number of tickets for each job.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–pe` *parallel_environment n[-[m]]|[-]m,...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Parallel programming environment (PE) to instantiate. The range descriptor behind the PE name specifies the number of parallel processes to be run. Sun Grid Engine will allocate the appropriate resources as available. The `sge_pe(5)` manual page contains information about the definition of PEs and about how to obtain a list of currently valid PEs. You can specify the PE name by using the wildcard character, `*`, thus the request `pvm*` will match any parallel environment with a name starting with the string, `pvm`. The range specification is a list of range expressions of the form *n-m* (*n* as well as *m* being positive non-zero integer numbers), where *m* is an abbreviation for `m-m`, *-m* is a short form for `1-m` and *n-* is an abbreviation for `n-infinity`. The range specification is processed as follows: The largest number of queues requested is checked first. If enough queues meeting the specified attribute list are available, all are allocated. The next smaller number of queues is checked next and so forth. If additional `–l` options are present, they restrict the set of eligible queues for the parallel job. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
`–q` *`queue,...`*	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Defines or redefines a list of queues which may be used to execute this job. This parameter has all the properties of a resource request and will be merged with requirements derived from the `–l` option described above. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–r y|n`	Available for `qsub` and `qalter` only. Identifies the ability of a job to be rerun or not. If the value of `–r` is `y`, rerun the job if the job was aborted without leaving a consistent exit state (this is typically the case if the node on which the job is running crashes). If `–r` is `n`, do not rerun the job under any circumstances. Interactive jobs submitted with `qsh` or `qlogin` are not re-runable. `qalter` allows changing this option even while the job executes.
`-sc` *variable[=value],...*	Available for `qsub`, `qsh`, `qrsh`, `qlogin` and `qalter` only. Sets the given *name/value* pairs as the job's context. *value* may be omitted. Sun Grid Engine replaces the job's previously defined context with the one given as the argument. Multiple `-ac`, `-dc`, and `-sc` options may be given. The order is important here. Contexts are a way to dynamically attach and remove meta-information to and from a job. The context variables are not passed to the job's execution context in its environment. `qalter` allows changing this option even while the job executes.
`–soft`	Available for `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter` only. Signifies that all resource requirements following in the command line will be soft requirements and are to be filled on an "as available" basis. As Sun Grid Engine scans the command line and script file for Sun Grid Engine options and parameters it builds a list of resources required by a job. All such resource requests are considered as absolutely essential for the job to commence. If the `–soft` option is encountered during the scan then all following resources are designated as "soft requirements" for execution, or "nice-to-have, but not essential". If the `–hard` flag (see above) is encountered at a later stage of the scan, all resource requests following it once again become "essential". The `–hard` and `–soft` options in effect act as "toggles" during the scan.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`-S` *[host:]pathname,...*	Available for `qsub`, `qsh`, `qlogin` and `qalter`. Specifies the interpreting shell for the job. Only one *pathname* component without a *host* specifier is valid and only one path name for a given host is allowed. Shell paths with host assignments define the interpreting shell for the job if the host is the execution host. The shell path without host specification is used if the execution host matches none of the hosts in the list. Furthermore, the *pathname* can be constructed with pseudo environment variables as described for the `-e` option above. In the case of `qsh` the specified shell path is used to execute the corresponding command interpreter in the `xterm(1)` (via its `-e` option) started on behalf of the interactive job. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–t n[-m[:s]]`	Available for `qsub` and `qalter` only. Submits a so called *Job Array*; i.e., an array of identical tasks being only differentiated by an index number and being treated by Sun Grid Engine almost like a series of jobs. The option argument to `–t` specifies the number of job array tasks and the index number which will be associated with the tasks. The index numbers will be exported to the job tasks via the environment variable *SGE_TASK_ID*. The task id range specified in the option argument may be a single number, a simple range of the form *n-m* or a range with a step size. Hence, the task id range specified by 2-10:2 would result in the task id indexes 2, 4, 6, 8, and 10, i.e. in a total of 5 identical tasks with the environment variable *SGE_TASK_ID* containing one of the 5 index numbers each. All job array tasks inherit the same resource requests and attribute definitions as specified in the `qsub` or `qalter` command line, except for the `–t` option. The tasks are scheduled independently and, provided enough resources, concurrently very much like separate jobs. However, a job array or a sub-array thereof can be accessed as a total by commands like `qmod(1)` or `qdel(1)`. See the corresponding manual pages for further detail. Job arrays are commonly used to execute the same type of operation on varying input data sets correlated with the task index number. The number of tasks in a job array is unlimited. `STDOUT` and `STDERR` of job array tasks will be written into different files with the default location: *<jobname>*`.['e'|'o']`*<job_id>*`'.'`*<task_id>* In order to change this default, the `–e` and `–o` options (see above) can be used together with the pseudo environment variables `$HOME`, `$USER`, `$JOB_ID`, `$JOB_NAME`, `$HOSTNAME`, and `$SGE_TASK_ID`. You can use the output redirection to divert the output of all tasks into the same file, but the result of this is undefined.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–u` *username,... | -uall*	Available for `qalter` only. Changes are only made on those jobs which were submitted by users specified in the list of *usernames*. For managers it is possible to use the `qalter -uall` command to modify all jobs of all users. If you use the `–u` or `–uall` switch, you may *not* specify an additional *job/task_id_list*.
`–v` *variable[=value],...*	Available for `qsub`, `qrsh`, `qsh`, `qlogin,qresub`, and `qalter`. Defines or redefines the environment variables to be exported to the execution context of the job. If the `–v` option is present, Sun Grid Engine will add the environment variables defined as arguments to the switch and, optionally, values of specified variables, to the execution context of the job. `qalter` allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
`–verbose`	Available only for `qrsh` and `qmake(1)`. Unlike `qsh` and `qlogin`, `qrsh` does not output any informational messages while establishing the session compliant with the standard `rsh(1)` and `rlogin(1)` system calls. If the option `-verbose` is set, `qrsh` behaves as verbose as the `qsh` and `qlogin` commands and outputs information about the process of establishing the `rsh(1)` or `rlogin(1)` session.
`–verify`	Available for `qsub`, `qrsh`, `qsh`, `qlogin`, `qresub`, and `qalter`. Does not submit a job but prints information on the job as being represented by the current command line and all pertinent external influences.
`–V`	Available for `qresub`, `qsub`, `qrsh`, `qsh`, `qlogin` and `qalter`. Specifies that all environment variables active within the `qsub` utility be exported to the context of the job.

TABLE 25 `submit` Command Options *(Continued)*

Option	Description
`–w e|w|n|v`	Available for `qsub`, `qrsh`, `qsh`, `qlogin`, `qresub`, and `qalter`. Specifies a validation level applied to the job to be submitted (`qsub`, `qlogin`, and `qsh`) or the specified queued job (`qalter`). The information displayed indicates whether the job possibly can be scheduled assuming an empty system with no other jobs. Resource requests exceeding the configured maximal thresholds or requesting unavailable resource attributes are possible causes for jobs to fail this validation. The specifiers `e`, `w`, `n`, and `v` define the following validation modes: `e` – Error: Jobs with invalid requests will be rejected; this is the default for `qrsh, qsh`, and `qlogin`. `w` – Warning: Only a warning will be displayed for invalid requests. `n` – None: Switches off validation; the default for `qalter` and `qsub`. `v` – Verify: Does not submit the job but prints extensive validation report. The necessary checks are performance consuming and hence the checking is switched off by default. The reasons for job requirements being invalid with respect to resource availability of queues are displayed in the `-w v` case using the format as described for the `qstat(1)` –F option (see description of Full Format in the Output Formats section of the `qstat(1)` manual page).
job/task_id_list	Specified by the following form: *job_id[.task_range][,job_id[.task_range],...]* If present, the *task_range* restricts the effect of the operation to the job array task range specified as suffix to the job id (see the `–t` option to `qsub(1)` for further details on job arrays). The task range specifier has the form *n[-m[:s]]*. The range may be a single number, a simple range of the form *n-m* or a range with a step size. Instead of *job/task_id_list* it is possible to use the keyword `all` to modify all jobs of the current user.

TABLE 25 submit Command Options *(Continued)*

Option	Description
scriptfile	Available for qsub only. The job's *scriptfile*. If not present or if the operand is the single-character string, -, qsub reads the script from standard input.
script_args	Available for qsub and qalter only. Arguments to the job. Not valid if the script is entered from standard input. qalter allows changing this option even while the job executes. The modified parameter will only be in effect after a restart or migration of the job, however.
xterm_args	Available for qsh only. Arguments to the xterm(1) executable, as defined in the configuration. For details, refer to sge_conf(5)).

Environment Variables

TABLE 26 lists the environment variables associated with submit.

TABLE 26 submit Environment Variables

Name of Variable	Description
SGE_ROOT	Specifies the location of the Sun Grid Engine standard configuration files.
SGE_CELL	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, qtcsh uses (in the order of precedence): • The name of the cell specified in the environment variable, SGE_CELL, if it is set. • The name of the default cell; e.g., default.
SGE_DEBUG_LEVEL	If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
COMMD_PORT	If set, specifies the TCP port on which sge_commd(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
COMMD_HOST	If set, specifies the host on which the particular sge_commd(8) to be used for Sun Grid Engine communication of the qsub, qsh, qlogin or qalter client resides. Per default the local host is used.

In addition to those environment variables specified to be exported to the job via the `-v` or the `-V` option (see above), `qsub`, `qsh`, and `qlogin` add the following variables with the indicated values, shown in TABLE 27, to the variable list.

TABLE 27 Additional `qsub`, `qsh`, and `qlogin` Environment Variables

Name of Variable	Description
`SGE_O_HOME`	Home directory of the submitting client.
`SGE_O_HOST`	Name of the host on which the submitting client is running.
`SGE_O_LOGName`	*LOGNAME* of the submitting client.
`SGE_O_MAIL`	*MAIL* of the submitting client.; this is the mail directory of the submitting client.
`SGE_O_PATH`	Executable search path of the submitting client.
`SGE_O_SHELL`	*SHELL* of the submitting client.
`SGE_O_TZ`	Time zone of the submitting client.
`SGE_O_WORKDIR`	Absolute path of the current working directory of the submitting client.

Sun Grid Engine software also sets additional variables into the job's environment, as listed in TABLE 28.

TABLE 28 Variables Set in Job Environment by `submit` Commands

Name of Variable	Description
`ARC`	The Sun Grid Engine architecture name of the node on which the job is running. The name is compiled-in into the `sge_execd(8)` binary.
`SGE_CKPT_ENV`	Specifies the checkpointing environment (as selected with the `-ckpt` option) under which a checkpointing job executes. Only set for checkpointing jobs.
`SGE_CKPT_DIR`	Only set for checkpointing jobs. Contains path *ckpt_dir* (see `checkpoint(5)` of the checkpoint interface).
`SGE_STDERR_PATH`	The path name of the file to which the standard error stream of the job is diverted. Commonly used for enhancing the output with error messages from *prolog*, *epilog*, parallel environment start/stop or checkpointing scripts.
`SGE_STDOUT_PATH`	The path name of the file to which the standard output stream of the job is diverted. Commonly used for enhancing the output with messages from *prolog*, *epilog*, parallel environment start/stop or checkpointing scripts.

TABLE 28 Variables Set in Job Environment by `submit` Commands

Name of Variable	Description
`SGE_JOB_SPOOL_DIR`	The directory used by `sge_shepherd(8)` to store job-related data during job execution. This directory is owned by root or by a Sun Grid Engine administrative account and commonly is not open for read or write access to regular users.
`SGE_TASK_ID`	The index number of the current job array task (see `-t` option above). This is a unique number in each job array and can be used to reference different input data records, for example. This environment variable is not set for non-array jobs.
`ENVIRONMENT`	The `ENVIRONMENT` variable is set to `BATCH` to identify that the job is being executed under Sun Grid Engine control.
`HOME`	The user's home directory path from the `passwd(5`*)* file.
`HOSTNAME`	The host name of the node on which the job is running.
`JOB_ID`	A unique identifier assigned by the `sge_qmaster(8)` when the job was submitted. The job ID is a decimal integer in the range 1 to 99999.
`JOB_NAME`	The job name; either `INTERACT` for interactive jobs or built from the `qsub` script file name, a period, and the digits of the job ID. This default may be overwritten by the `-N` option.
`LAST_HOST`	The name of the preceding host in case of migration of a checkpointing job.
`LOGNAME`	The user's login name from the `passwd(5)` file.
`NHOSTS`	The number of hosts in use by a parallel job.
`NQUEUES`	The number of queues allocated for the job (always 1 for serial jobs).
`NSLOTS`	The number of queue slots in use by a parallel job.
`PATH`	A default shell search path of: `/usr/local/bin:/usr/ucb:/bin:/usr/bin`
`PE`	The parallel environment under which the job executes (for parallel jobs only).
`PE_HOSTFILE`	The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Sun Grid Engine. See the description of the *$pe_hostfile* parameter in `sge_pe(5)` for details on the format of this file. The environment variable is only available for parallel jobs.
`QUEUE`	The name of the queue in which the job is running.
`REQUEST`	Available for batch jobs only. The request name of a job as specified with the `-N` switch (see above) or taken as the name of the job script file.

TABLE 28 Variables Set in Job Environment by `submit` Commands

Name of Variable	Description
`RESTARTED`	This variable is set to `1` if a job was restarted either after a system crash or after a migration in case of a checkpointing job. The variable has the value `0` otherwise.
`SHELL`	The user's login shell from the `passwd(5)` file. Note: This is not necessarily the shell in use for the job.
`TMPDIR`	The absolute path to the job's temporary working directory.
`TMP`	The same as `TMPDIR`; provided for compatibility with NQS.
`TZ`	The time zone variable imported from `sge_execd(8)` if set.
`USER`	The user's login name from the `passwd(5)` file.

Restrictions

There is no controlling terminal for batch jobs under Sun Grid Engine and any tests or actions on a controlling terminal will fail. If these operations are in your `.login` or `.cshrc` file, they will possibly cause your job to abort.

Insert the following test before any commands that are not pertinent to batch jobs in your`.login` file.

```
if ( $?JOB_NAME) then
echo "Sun Grid Engine spooled job"
exit 0
endif
```

Don't forget to set your shell's search path in your shell startup before this code.

Exit Status

The following exit values are returned.

- `0` – Operation was executed successfully

- `25` – It was not possible to register a new job according to the configured *max_u_jobs* or *max_jobs* limit. Additional information may be found in `sge_conf(5)`.
- `>0` – Error occured.

Examples

The following is the simplest form of a Sun Grid Engine script file.

```
#!/bin/csh
a.out
```

The next example is a more complex Sun Grid Engine script.

```
#!/bin/csh

# Which account to be charged cpu time
#$ -A santa_claus

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# to run I want 6 or more parallel processes
# under the PE pvm. the processes require
# 128M of memory
#$ -pe pvm 6- -l mem=128

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo

# Send mail to these users
#$ -M santa@heaven,claus@heaven

# Mail at beginning/end/on suspension
#$ -m bes

# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR

# The job is located in the current
# working directory.
#$ -cwd

a.out
```

Files

- `STDOUT` of job #JID – `$REQUEST.oJID[.`*taskid*`]`
- `STDERR` of job – `$REQUEST.eJID[.`*taskid*`]`
- `STDOUT` of par. env. of job – `$REQUEST.poJID[.`*taskid*`]`
- `STDERR` of par. env. of job – `$REQUEST.peJID[.`*taskid*`]`
- Hosts file of par. env. of job – `$REQUEST.hostsJID[.`*taskid*`]`
- `cwd` path aliases – `$cwd/.sge_aliases`
- `cwd` default request – `$cwd/.sge_request`
- User path aliases – `$HOME/.sge_aliases`
- User default request – `$HOME/.sge_request`
- Cluster path aliases – <*sge_root*>`/`<*cell*>`/common/.sge_aliases`
- Cluster default request – <*sge_root*>`/`<*cell*>`/common/.sge_request`
- Sun Grid Engine master host file – <*sge_root*>`/`<*cell*>`/common/act_qmaster`

See Also

`sge_intro(1), qconf(1), qdel(1), qhold(1), qmod(1), qrls(1), qstat(1), accounting(5), sge_aliases(5), sge_conf(5), sge_request(5), sge_pe(5), complex(5)`

Copyright

If configured correspondingly, `qrsh` and `qlogin` contain portions of the `rsh`, `rshd`, `telnet` and `telnetd` code copyrighted by The Regents of the University of California. Therefore, the following note applies with respect to `qrsh` and `qlogin`: This product includes software developed by the University of California, Berkeley and its contributors.

See `sge_intro(1)` as well as the information provided in <*sge_root*>`/3rd_party/qrsh` and <*sge_root*>`/3rd_party/qlogin` for a statement of further rights and permissions.

access_list(5)

Name

access_list – Sun Grid Engine access list file format

Description

Access lists are used in Sun Grid Engine products to define access permissions of users to queues (see queue_conf(5)) or parallel environments (see sge_pe(5)). A list of currently configured access lists can be displayed via the qconf(1) -sul option. The contents of each enlisted access list can shown via the -su switch. The output follows the access_list format description. New access lists can be created and existing can be modified via the -au and -du options to qconf(1).

Format

Each user or UNIX user group appears in a single lines. Only symbolic names are allowed. A group is differentiated from a user name by prefixing the group name with a ’@’ sign.

See Also

sge_intro(1), qconf(1), sge_pe(5), queue_conf(5)

Copyright

See sge_intro(1) for a full statement of rights and permissions.

accounting(5)

Name

accounting – Sun Grid Engine accounting file format

Description

An accounting record is written to the Sun Grid Engine accounting file for each job having finished. The accounting file is processed by qacct(1) to derive accounting statistics.

Format

Each job is represented by a line in the accounting file. Empty lines and lines that contain one character or less are ignored. Accounting record entries are separated by colon (:) signs. In the following the entries are denoted in their order of appearance.

- qname – This is the name of the queue in which the job has run.
- hostname – This is the name of the execution host.
- group – This is the effective group id of the job owner when executing the job.
- owner – This is the owner of the Sun Grid Engine job.
- job_name – This is the job name.
- job_number – This is the job identifier or job number.
- account – This is the an account string as specified by the qsub(1) or qalter(1) –A option.
- priority – This is the priority value assigned to the job corresponding to the priority parameter in the queue configuration (see queue_conf(5)).
- submission_time – This is the submission time in seconds (since epoch format).
- start_time – This is the start time in seconds (since epoch format).
- end_time – This is the end time in seconds (since epoch format).

- failed – This indicates the problem that occurred in case a job could not be started on the execution host (e.g., because the owner of the job did not have a valid account on that machine). If Sun Grid Engine software tries to start a job multiple times, this may lead to multiple entries in the accounting file corresponding to the same job ID.
- exit_status – This is the exit status of the job script (or Sun Grid Engine specific status in case of certain error conditions).
- ru_wallclock – This is the difference between end_time and start_time (see above).

  The remainder of the accounting entries in this category follows the contents of the standard UNIX rusage structure as described in getrusage(2). The following entries are provided.

  - ru_utime
  - ru_stime
  - ru_maxrss
  - ru_ixrss
  - ru_ismrss
  - ru_idrss
  - ru_isrss
  - ru_minflt
  - ru_majflt
  - ru_nswap
  - ru_inblock
  - ru_oublock
  - ru_msgsnd
  - ru_msgrcv
  - ru_nsignals
  - ru_nvcsw
  - ru_nivcsw
- project – This is the project that was assigned to the job. Projects are only supported in case of a Sun Grid Engine, Enterprise Edition system.
- department – This is the department that was assigned to the job. Departments are only supported in case of a Sun Grid Engine, Enterprise Edition system.
- granted_pe – This is the parallel environment that was selected for that job.
- slots – This is the number of slots that were dispatched to the job by the scheduler.
- task_number – This is the job array task index number.
- cpu – This is the cpu time usage in seconds.
- mem – This is the integral memory usage in Gbytes seconds.
- io – This is the amount of data transferred in input/output operations.
- category – This is a string specifying the job category.
- iow – This is the I/O wait time, in seconds.

- `pe_taskid` – If this identifier is set the task was part of a parallel job and was passed to the Sun Grid Engine, Enterprise Edition system via the `qrsh -inherit` interface.
- `maxvem` – This is the maximum `vmem` size, in bytes.

See Also

`sge_intro(1)`, `qacct(1)`, `qalter(1)`, `qsub(1)`, `getrusage(2)`, `queue_conf(5)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

calendar_conf(5)

Name

`calendar_conf` – Sun Grid Engine calendar configuration file format

Description

`calendar_conf` reflects the format of the Sun Grid Engine calendar configuration. The definition of calendars is used to specify “on duty” and “off duty” time periods for Sun Grid Engine queues on a time of day, day of week or day of year basis. Various calendars can be implemented and the appropriate calendar definition for a certain class of jobs can be attached to a queue.

`calendar_conf` entries can be added, modified and displayed with the `-Acal`, `-acal`, `-Mcal`, `-mcal`, `-scal` and `-scall` options to `qconf(1)` or with the calendar configuration dialog of the graphical user interface `qmon(1)`. The format of the calendar configuration entries is defined as follows:

Format

- `calendar_name` – This is the name of the calendar to be used when attaching it to queues or when administering the calendar definition.
- `year` – This is the queue status definition on a day of the year basis. This field generally will specify on which days of a year (and optionally at which times on those days) a queue, to which the calendar is attached, will change to a certain state. The syntax of the year field is defined as follows:

  `year:=`

  {year_day_range_list[=daytime_range_list][=state]

  |[year_day_range_list=]daytime_range_list[=state]

  |[year_day_range_list=][daytime_range_list=]state} ...

  Where:

  - At least one of *year_day_range_list*, *daytime_range_list* and *state* always have to be present.
  - Every day in the year is assumed if *year_day_range_list* is omitted.
  - All day long is assumed if *daytime_range_list* is omitted.
  - Switching the queue to "off" (i.e. disabling it) is assumed if *state* is omitted.
  - The queue is assumed to be enabled for days neither referenced implicitly (by omitting the *year_day_range_list*) nor explicitly.

  And the syntactical components are defined as follows:

  year_day_range_list := *{yearday-yearday|yearday},...*

  daytime_range_list := *hour[:minute][:second]-hour[:minute][:second],...*

  state := {`on|off|suspended`}

  year_day := *month_day.month.year*

  month_day := {`1|2|...|31`}

  month := {`jan|feb|...|dec|1|2|...|12`}

  year := {`1970|1971|...|2037`}

- `week` – This is the queue status definition on a day of the week basis. This field generally will specify on which days of a week (and optionally at which times on those days) a queue, to which the calendar is attached, will change to a certain state.

  The syntax of the `week` field is as follows:

  `week:=`

  {*week_day_range_list*[=*daytime_range_list*][=*state*]

  |[*week_day_range_list*=]*daytime_range_list*[=*state*]

|[*week_day_range_list*=][*daytime_range_list*=]*state*} ...

Where

- At least one of *week_day_range_list*, *daytime_range_list* and *state* always have to be present.
- Every day in the week is assumed if *week_day_range_list* is omitted.
- Syntax and semantics of *daytime_range_list* and *state* are identical to the definition given for the year field above.
- The queue is assumed to be enabled for days neither referenced implicitly (by omitting the *week_day_range_list*) nor explicitly.

And where *week_day_range_list* is defined as

week_day_range_list := {*weekday-weekday*|*weekday*},...

week_day := {`mon|tue|wed|thu|fri|sat|sun`}

Semantics

Successive entries to the *year* and *week* fields (separated by blanks) are combined in compliance with the following rule:

- `off` – Areas are overridden by overlapping `on` and `suspended` areas.

  Hence, an entry of the form, `week 12-18 tue=13-17=on` means that queues referencing the corresponding calendar are disabled the entire week with the exception of Tuesday between 13.00-17.00 where the queues are available.

Examples

(The following examples are contained in the directory, $*sge_root*`/util/resources/calendars`).

- Night, weekend, and public holiday calendar – On public holidays, "night" queues are explicitly enabled. On working days queues are disabled between 6.00 and 20.00. Saturday and Sunday are implicitly handled as enabled times:.

```
calendar_namenight
year
1.1.1999,6.1.1999,28.3.1999,30.3.1999-31.3.1999,18.5.1999-19.5.1999,
3.10.1999,25.12.1999,26.12.1999=on
weekmon-fri=6-20
```

- Day calendar – On public holidays, “day”-queues are disabled. On working days such queues are closed during the night between 20.00 and 6.00, i.e. the queues are also closed on Monday from 0.00 to 6.00 and on Friday from 20.00 to 24.00. On Saturday and Sunday the queues are disabled.

```
calendar_nameday
year
1.1.1999,6.1.1999,28.3.1999,30.3.1999-31.3.1999,18.5.1999-19.5.1999,
3.10.1999,25.12.1999,26.12.1999
weekmon-fri=20-6 sat-sun
```

- Night, weekend and public holiday calendar with suspension – Essentially the same scenario as the first example, but queues are suspended instead of switching them “off.”

```
calendar_namenight_s
year
1.1.1999,6.1.1999,28.3.1999,30.3.1999-31.3.1999,18.5.1999-19.5.1999,
3.10.1999,25.12.1999,26.12.1999=on
weekmon-fri=6-20=suspended
```

- Day calendar with suspension – Essentially the same scenario as the second example, but queues are suspended instead of switching them “off.”

```
calendar_nameday_s
year
1.1.1999,6.1.1999,28.3.1999,30.3.1999-31.3.1999,18.5.1999-19.5.1999,
3.10.1999,25.12.1999,26.12.1999=suspended
week         mon-fri=206=suspended sat-sun=suspended
```

See Also

`sge_intro(1)`, `qconf(1)`, `queue_conf(5)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

checkpoint(5)

Name

checkpoint – Sun Grid Engine checkpointing environment configuration file format

Description

Checkpointing is a facility to save the complete status of an executing program or job and to restore and restart from this so called checkpoint at a later point of time if the original program or job was halted, e.g. through a system crash.

Sun Grid Engine provides various levels of checkpointing support (see `sge_ckpt(1)`). The checkpointing environment described here is a means to configure the different types of checkpointing in use for your Sun Grid Engine cluster or parts thereof. For that purpose you can define the operations which have to be executed in initiating a checkpoint generation, a migration of a checkpoint to another host or a restart of a checkpointed application as well as the list of queues which are eligible for a checkpointing method.

Supporting different operating systems may easily force Sun Grid Engine to introduce operating system dependencies for the configuration of the checkpointing configuration file and updates of the supported operating system versions may lead to frequently changing implementation details. Refer to the file, <*sge_root*>`/doc/checkpointing.asc` for more information.

Use the `–ackpt`, `–dckpt`, `–mckpt` or `–sckpt` options to the `qconf(1)` command to manipulate checkpointing environments from the command-line or use the corresponding `qmon(1)` dialogue for X-Windows based interactive configuration.

Format

The format of a `checkpoint` file is defined as follows:

- `ckpt_name` – This is the name of the checkpointing environment. To be used in the `qsub(1)` –ckpt switch or for the `qconf(1)` options mentioned above.
- `interface` – This is the type of checkpointing to be used. Currently, the following types are valid:
  - `hibernator` – This is the Hibernator kernel level checkpointing is interfaced.

    - `cpr` – This is the SGI kernel level checkpointing is used.
    - `cray-ckpt` – This is the Cray kernel level checkpointing is assumed.
    - `transparent` – The Sun Grid Engine system assumes that the jobs submitted with reference to this checkpointing interface use a checkpointing library such as provided by the public domain package *Condor*.
    - `userdefined` – The Sun Grid Engine system assumes that the jobs submitted with reference to this checkpointing interface perform their private checkpointing method.
    - `application-level` – This uses all of the interface commands configured in the checkpointing object like in the case of one of the kernel level checkpointing interfaces (`cpr`, `cray-ckpt`, etc.) except for the *restart_command* (see below), which is not used (even if it is configured) but the job script is invoked in case of a restart instead.
- `queue_list` – This is a comma-separated list of queues to which parallel jobs belonging to this parallel environment have access to.
- `ckpt_command` – This is command-line type command string to be executed by Sun Grid Engine in order to initiate a checkpoint.
- `migr_command` – This is a command-line type command string to be executed by Sun Grid Engine during a migration of a checkpointing job from one host to another.
- `restart_command` – This is a command-line type command string to be executed by Sun Grid Engine when restarting a previously checkpointed application.
- `clean_command` – This is a command-line type command string to be executed by Sun Grid Engine in order to cleanup after a checkpointed application has finished.
- `ckpt_dir` – This is a file system location to which checkpoints of potentially considerable size should be stored.
- `queue_list` – This contains a comma- or blank-separated list of queue names which are eligible for a job if the checkpointing environment was specified at the submission of the job.
- `ckpt_signal` – This is a UNIX signal to be sent to a job by Sun Grid Engine to initiate a checkpoint generation. The value for this field can either be a symbolic name from the list produced by the `-l` option of the `kill(1)` command or an integer number which must be a valid signal on the systems used for checkpointing.
- `when` – This establishes the points of time when checkpoints are expected to be generated. Valid values for this parameter are composed by the letters `s`, `m`, `x`, and `r`, and any combinations thereof without any separating character in between. The same letters are allowed for the `-c` option of the `qsub(1)` command which will overwrite the definitions in the used checkpointing environment. The meaning of the letters is defined as follows:

- `s` – With this letter, a job is checkpointed, aborted and if possible migrated if the corresponding `sge_execd(8)` is shut down on the job's machine.
- `m` – With this letter, checkpoints are generated periodically at the *min_cpu_interval* interval defined by the queue (see `queue_conf(5)`) in which a job executes.
- `x` – With this letter, a job is checkpointed, aborted and if possible migrated as soon as the job gets suspended (manually as well as automatically).
- `r` – With this letter, a job will be rescheduled (not checkpointed) when the host on which the job currently runs went into unknown state and the time interval, `reschedule unknown` (see `sge_conf(5)`), defined in the global/local cluster configuration will be exceeded.

Restrictions

Note – The functionality of any checkpointing, migration or restart procedures provided by default with the Sun Grid Engine distribution, as well as the way they are invoked in the *ckpt_command*, *migr_command* or *restart_command* parameters of any default checkpointing environments, should not be changed or otherwise the functionality remains the full responsibility of the administrator configuring the checkpointing environment. Sun Grid Engine software will just invoke these procedures and evaluate their exit status. If the procedures do not perform their tasks properly or are not invoked in a proper fashion, the checkpointing mechanism may behave unexpectedly. The Sun Grid Engine system has no means to detect this.

See Also

`sge_intro(1)`, `sge_ckpt(1)`, `qconf(1)`, `qmod(1)`, `qsub(1)`, `sge_execd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_request(5)

Name

sge_request – Sun Grid Engine default request definition file format

Description

sge_request reflects the format of the files to define default request profiles. If available, default request files are read and processed during job submission before any submit options embedded in the job script and before any options in the qsub(1) or qsh(1) command-line are considered. Thus, the command-line and embedded script options may overwrite the settings in the default request files (see qsub(1) or qsh(1) for details).

There is a cluster global, a user private and a working directory local default request definition file. The working directory local default request file has the highest precedence and is followed by the user private and then the cluster global default request file.

Note – The -clear option to qsub(1) or qsh(1) can be used to discard any previous settings at any time in a default request file, in the embedded script flags or in a qsub(1) or qsh(1) command-line option.

The format of the default request definition files follows.

- The default request files may contain an arbitrary number of lines. Blank lines and lines with a '#' sign in the first column are skipped.
- Each line not to be skipped may contain any qsub(1) option as described in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*. More than one option per line is allowed. The batch script file and argument options to the batch script are not considered as qsub(1) options and thus are not allowed in a default request file.

Examples

The following is a simple example of a default request definition file:

```
# Default Requests File
# request arch to be sun4 and a CPU-time of 5hr
-l arch=sun4,s_cpu=5:0:0
# don't restart the job in case of system crashes
-r n
```

Having defined a default request definition file like this and submitting a job as follows:

```
qsub test.sh
```

would have precisely the same effect as if the job was submitted with:

```
sub -l arch=sun4,s_cpu=5:0:0 -r n test.sh
```

Files

- Global defaults file – *<sge_root>/<cell>*`/common/sge_request`
- User private defaults file – *$HOME/*`.sge_request`
- `cwd` directory defaults file – *$cwd*`/.sge_request`

See Also

`sge_intro(1)`, `qsh(1)`, `qsub(1)`, *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide, Sun Grid Engine 5.3 Administration and User's Guide*

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_aliases(5)

Name

`sge_aliases` – Sun Grid Engine path aliases file format

Description

The Sun Grid Engine path aliasing facility provides administrators and users with the means to reflect complicated and in-homogeneous file system structures in distributed environments (such as user home directories mounted under different paths on different hosts) and to ensure that Sun Grid Engine is able to locate the appropriate working directories for executing batch jobs.

There is a system global path aliasing file and a user local file . `sge_aliases` defines the format of both:

- Blank lines and lines with a '#' sign in the first column are skipped.
- Each line other than a blank line or a line lead by '#' has to contain four strings separated by any number of blanks or tabs.
- The first string specifies a source-path, the second a submit-host, the third an execution-host and the fourth the source-path replacement.
- Both the submit- and the execution-host entries may consist of only a '*' sign which matches any host.

If the `-cwd` flag (and only if – otherwise the user's home directory on the execution host is selected to execute the job) to `qsub(1)` was specified, the path aliasing mechanism is activated and the files are processed as follows:

- After `qsub(1)` has retrieved the physical current working directory path, the cluster global path aliasing file is read if present. The user path aliases file is read afterwards as if it were appended to the global file.
- Lines not to be skipped are read from the top of the file one by one while the translations specified by those lines are stored if necessary.
- A translation is stored only if the submit-host entry matches the host `qsub(1)` is executed on and if the source-path forms the initial part either of the current working directory or of the source-path replacements already stored.
- As soon as both files are read the stored path aliasing information is passed along with the submitted job.

- On the execution host, the aliasing information will be evaluated. The leading part of the current working directory will be replaced by the source-path replacement if the execution-host entry of the path alias matches the executing host.

Note – The current working directory string will be changed in this case and subsequent path aliases must match the replaced working directory path to be applied.

Examples

The following is a simple example of a path aliasing file resolving problems with in-homogeneous paths if `automount(8)` is used:

```
# Path Aliasing File
# src-pathsub-hostexec-hostreplacement
/tmp_mnt/**/
# replaces any occurrence of /tmp_mnt/ by /
# if submitting or executing on any host.
# Thus paths on nfs server and clients are the same
```

Files

- Global aliases file – *<sge_root>/<cell>*`/common/sge_aliases`
- User local aliases file – *$HOME*`/.sge_aliases`

See Also

`sge_intro(1)`, `qsub(1)`, *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide, Sun Grid Engine 5.3 Administration and User's Guide*

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_conf(5)

Name

sge_conf – Sun Grid Engine configuration files

Description

sge_conf defines the global and local Sun Grid Engine configurations and can be shown and modified by qconf(1) using the -sconf and -mconf options. Only root or the cluster administrator may modify sge_conf.

At its initial startup, sge_qmaster(8) checks to see if a valid Sun Grid Engine configuration is available at a well known location in the Sun Grid Engine internal directory hierarchy. If so, it loads that configuration information and proceeds. If not, sge_qmaster(8) writes a generic configuration containing default values to that same location. The Sun Grid Engine execution daemons sge_execd(8) upon startup retrieve their configuration from sge_qmaster(8).

The actual configuration for both sge_qmaster(8) and sge_execd(8) is a superposition of a so called *global* configuration and a *local* configuration being pertinent for the host on which a master or execution daemon resides. If a local configuration is available, its entries overwrite the corresponding entries of the global configuration.

Note – The local configuration does not have to contain all valid configuration entries, but only those which need to be modified against the global entries.

Format

The paragraphs that follow provide brief descriptions of the individual parameters that compose the global and local configurations for a Sun Grid Engine cluster.

- *qmaster_spool_dir* – This is the location where the master spool directory resides. Only the sge_qmaster(8) and sge_shadowd(8) need to have access to this directory. It needs read/write permission for root, however. The master spool directory—in particular the jobs directory and the message log file—may become quite large depending on the size of the cluster and the number of jobs. Be sure to allocate enough disk space and regularly clean off the log files, e.g. via a `cron(8)` job.

Because it is a parameter set at installation time, changing *qmaster_spool_dir* in a running system is not supported.

The default location for the master spool directory is: *<sge_root>*/*<cell>*`/spool/qmaster`

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *execd_spool_dir* – This is the execution daemon spool directory path. Again, a feasible spool directory requires read/write access permission for root. The entry in the global configuration for this parameter can be overwritten by execution host local configurations, i.e. each `sge_execd(8)` may have a private spool directory with a different path, in which case it needs to provide read/write permission for the root account of the corresponding execution host only.

  Under *execd_spool_dir* a directory named corresponding to the unqualified hostname of the execution host is opened and contains all information spooled to disk. Thus, it is possible for the *execd_spool_dirs* of all execution hosts to physically reference the same directory path (the root access restrictions mentioned above need to be met, however).

  Because it is a parameter set at installation time, changing *execd_spool_dir* in a running system is not supported.

  The default location for the execution daemon spool directory is *<sge_root>*/*<cell>*`/spool`.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *binary_path* – This is the directory path where the Sun Grid Engine binaries reside. It is used within Sun Grid Engine components to locate and star tup other Sun Grid Engine programs.

  The path name given here is searched for binaries as well as any directory below with a directory name equal to the current operating system architecture. Therefore, `/usr/SGE/bin` will work for all architectures, if the corresponding binaries are located in subdirectories named `aix43`, `cray`, `glinux`, `hp10`, `irix6`, `osf4`, `solaris`, etc.

  Each `sge_execd(8)` may have its private binary path. Changing the binary path will have immediate effect for `sge_execd(8)`.

  The default location for the binary path is: *<sge_root>*`/bin`

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *mailer* – This is the absolute pathname to the electronic mail delivery agent on your system. It must accept the following syntax:

`mailer -s` *<subject-of-mail-message> <recipient>*

Each `sge_execd(8)` may use a private mail agent. Changing *mailer* will take immediate effect.

The default for *mailer* depends on the operating system of the host on which the Sun Grid Engine master installation was run. Common values are `/bin/mail` or `/usr/bin/Mail`.

The global configuration entry for this value may be overwritten by the execution host local configuration.

- *xterm* – This is the absolute path name to the X Window System terminal emulator, `xterm(1)`.

  Each `sge_execd(8)` may use a private mail agent. Changing *xterm* will take immediate effect.

  The default for *xterm* is: `/usr/bin/X11/xterm`

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *load_sensor* – This is a comma-separated list of executable shell script paths or programs to be started by `sge_execd(8)` and to be used in order to retrieve site configurable load information (e.g., free space on a certain disk partition).

  Each `sge_execd(8)` may use a set of private *load_sensor* programs or scripts. Changing *load_sensor* will take effect after two load report intervals (see *load_report_time*). A load sensor will be restarted automatically if the file modification time of the load sensor executable changes.

  The global configuration entry for this value may be over-written by the execution host local configuration.

  In addition to the load sensors configured via *load_sensor*, `sge_execd(8)` searches for an executable file named `qloadsensor` in the execution host's Sun Grid Engine binary directory path. If such a file is found, it is treated like the configurable load sensors defined in *load_sensor*. This facility is intended for pre-installing a default load sensor.

- *prolog* – This is the executable path of a shell script that is started before execution of Sun Grid Engine jobs with the same environment setting as that for the Sun Grid Engine jobs to be started afterwards. An optional prefix "user@" specifies the user under which this procedure is to be started. This procedure is intended as a means for the Sun Grid Engine administrator to automate the execution of general site specific tasks like the preparation of temporary file systems with the need for the same context information as the job. Each `sge_execd(8)` may use a private *prolog* script. Correspondingly, the execution host local configurations is can be overwritten by the queue configuration (see `queue_conf(5)`). Changing *prolog* will take immediate effect.

Note – *prolog* is executed *exactly* as the job script. Therefore, all implications described under the parameters *shell_start_mode* and *login_shells* below apply.

The default for *prolog* is the special value `NONE`, which prevents from execution of a prologue script.

The following special variables being expanded at runtime can be used (besides any other strings which have to be interpreted by the procedure) to constitute a command line:

- *$host* – This is the name of the host on which the prolog or epilog procedures are started.
- *$job_owner* – This is the user name of the job owner.
- *$job_id* – This is the Sun Grid Engine system's unique job identification number.
- *$job_name* – This is the name of the job.
- *$processors* – The *processors* string as contained in the queue configuration (see `queue_conf(5)`) of the master queue (the queue in which the *prolog* and *epilog* procedures are started).
- *$queue* – The master queue, i.e. the queue in which the *prolog* and *epilog* procedures are started. The global configuration entry for this value may be overwritten by the execution host local configuration.

- *epilog* – This is the executable path of a shell script that is started after execution of Sun Grid Engine jobs with the same environment setting as that for the Sun Grid Engine jobs that has just completed. An optional prefix `user@` specifies the user under which this procedure is to be started. This procedure is intended as a means for the Sun Grid Engine administrator to automate the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job. Each `sge_execd(8)` may use a private *epilog* script. Correspondingly, the execution host local configurations is can be overwritten by the queue configuration (see `queue_conf(5)`). Changing *epilog* will take immediate effect.

Note – *epilog* is executed *exactly* as the job script. Therefore, all implications described under the parameters *shell_start_mode* and *login_shells* below apply.

The default for *epilog* is the special value `NONE`, which prevents from execution of a epilogue script. The same special variables as for *prolog* can be used to constitute a command line.

The global configuration entry for this value may be overwritten by the execution host local configuration.

- *shell_start_mode* – This parameter defines the mechanisms which are used to actually invoke the job scripts on the execution hosts. The following values are recognized:
  - `unix_behavior` – If a user starts a job shell script under UNIX interactively by invoking it just with the script name the operating system's executable loader uses the information provided in a comment such as '#!/bin/csh' in the first line of the script to detect which command interpreter to start to interpret the script. This mechanism is used by Sun Grid Engine when starting jobs if `unix_behavior` is defined as *shell_start_mode*.
  - `posix_compliant` – POSIX does not consider first script line comments such a '#!/bin/csh' as being significant. The POSIX standard for batch queuing systems (P1003.2d) therefore requires a compliant queuing system to ignore such lines but to use user specified or configured default command interpreters instead. Thus, if *shell_start_mode* is set to `posix_compliant` Sun Grid Engine will either use the command interpreter indicated by the `-S` option of the `qsub(1)` command or the *shell* parameter of the queue to be used (see `queue_conf(5)` for details).
  - `script_from_stdin` – Setting the *shell_start_mode* parameter either to `posix_compliant` or `unix_behavior` requires you to set the umask in use for `sge_execd(8)` such that every user has read access to the active_jobs directory in the spool directory of the corresponding execution daemon. In case you have *prolog* and *epilog* scripts configured, they also need to be readable by any user who may execute jobs.

    If this violates your site's security policies you may want to set *shell_start_mode* to `script_from_stdin`. This will force Sun Grid Engine to open the job script as well as the epilogue and prologue scripts for reading into STDIN as root (if `sge_execd(8)` was started as root) before changing to the job owner's user account. The script is then fed into the STDIN stream of the command interpreter indicated by the `-S` option of the `qsub(1)` command or the *shell* parameter of the queue to be used (see `queue_conf(5)` for details).

    Thus setting *shell_start_mode* to `script_from_stdin` also implies `posix_compliant` behavior.

Note – Feeding scripts into the `STDIN` stream of a command interpreter may cause trouble if commands like `rsh(1)` are invoked inside a job script as they also process the `STDIN` stream of the command interpreter. These problems can usually be resolved by redirecting the `STDIN` channel of those commands to come from `/dev/null` (e.g. `rsh` *host date* `< /dev/null`).

Note – Any command-line options associated with the job are passed to the executing shell. The shell will only forward them to the job if they are not recognized as valid shell options.

Changes to *shell_start_mode* will take immediate effect. The default for *shell_start_mode* is `posix_compliant`.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *login_shells* – UNIX command interpreters like the Bourne-Shell (see `sh(1)`) or the C-Shell (see `csh(1)`) can be used by Sun Grid Engine to start job scripts. The command interpreters can either be started as login-shells (i.e. all system and user default resource files like .login or .profile will be executed when the command interpreter is started and the environment for the job will be set up as if the user has just logged in) or just for command execution (i.e. only shell specific resource files like .cshrc will be executed and a minimal default environment is set up by Sun Grid Engine – see `qsub(1)`). The parameter *login_shells* contains a comma separated list of the executable names of the command interpreters to be started as login-shells.

  Changes to *login_shells* will take immediate effect. The default for *login_shells* is `sh,csh,tcsh,ksh`.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *min_uid* – This parameter places a lower bound on user IDs that may use the cluster. Users whose user ID (as returned by `getpwnam(3)`) is less than *min_uid* will not be allowed to run jobs on the cluster.

  Changes to *min_uid* will take immediate effect. The default for *min_uid* is 0.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *min_gid* – This parameter sets the lower bound on group IDs that may use the cluster. Users whose default group ID (as returned by `getpwnam(3)`) is less than *min_gid* will not be allowed to run jobs on the cluster.

  Changes to *min_gid* will take immediate effect. The default for *min_gid* is 0.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *user_lists* – This parameter contains a comma separated list of so called user access lists as described in `access_list(5)`. Each user contained in at least one of the enlisted access lists has access to the cluster. If the *user_lists* parameter is set to NONE (the default) any user has access being not explicitly excluded via the *xuser_lists* parameter described below. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to the cluster.

  Changes to *user_lists* will take immediate effect

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *xuser_lists* – This parameter contains a comma separated list of so called user access lists as described in `access_list(5)`. Each user contained in at least one of the enlisted access lists is denied access to the cluster. If the *xuser_lists* parameter is set to NONE (the default) any user has access. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* (see above) the user is denied access to the cluster.

  Changes to *xuser_lists* will take immediate effect

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *administrator_mail* – This parameter specifies a comma separated list of the electronic mail address(es) of the cluster administrator(s) to whom internally-generated problem reports are sent. The mail address format depends on your electronic mail system and how it is configured; consult your system's configuration guide for more information.

  Changing *administrator_mail* takes immediate effect. The default for *administrator_mail* is an empty mail list.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *projects* – This parameter is only available for Sun Grid Engine, Enterprise Edition systems. It is not present in Sun Grid Engine.

  The *projects* list contains all projects that are granted access to the Sun Grid Engine, Enterprise Edition system. Users belonging to none of these projects cannot use the Sun Grid Engine, Enterprise Edition system. If users belong to projects in the *projects* list and the *xprojects* list (see below), they also cannot use the system.

  Changing *projects* takes immediate effect. The default for *projects* is none.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *xprojects* – This parameter is only available for Sun Grid Engine, Enterprise Edition systems. It is not present in Sun Grid Engine.

  The *xprojects* list contains all projects which are granted access to Sun Grid Engine, Enterprise Edition. User belonging to one of these projects cannot use Sun Grid Engine, Enterprise Edition. If users belong to projects in the *projects* list (see above) and the *xprojects* list, they also cannot use the system.

  Changing *xprojects* takes immediate effect. The default for *xprojects* is none.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *load_report_time* – System load is reported periodically by the execution daemons to `sge_qmaster(8)`. The parameter *load_report_time* defines the time interval between load reports.

Each `sge_execd(8)` may use a different load report time. Changing *load_report_time* will take immediate effect.

Note – Be careful when modifying *load_report_time*. Reporting load too frequently might block `sge_qmaster(8)`, especially if the number of execution hosts is large. Moreover, since the system load typically increases and decreases smoothly, frequent load reports hardly offer any benefit.

The default for *load_report_time* is 40 seconds.

The global configuration entry for this value may be overwritten by the execution host local configuration.

- *reschedule_unknown* – Determines whether jobs on hosts in `unknown` state are rescheduled and thus sent to other hosts. Hosts are registered as `unknown` if `sge_master(8)` cannot establish contact to the `sge_execd(8)` on those hosts (see *max_unheard*). Likely reasons are a breakdown of the host or a breakdown of the network connection in between, but also `sge_execd(8)` may not be executing on such hosts.

  In any case, Sun Grid Engine can reschedule jobs running on such hosts to another system. *reschedule_unknown* controls the time which Sun Grid Engine will wait before jobs are rescheduled after a host became `unknown`. The time format specification is *hh:mm:ss*. If the special value `00:00:00` is set, then jobs will not be rescheduled from this host.

  Rescheduling is only initiated for jobs which have activated the rerun flag (see the `-r y` option of `qsub(1)` and the `rerun` option of `queue_conf(5)`). Parallel jobs are only rescheduled if the host on which their master task executes is in `unknown` state. Checkpointing jobs will only be rescheduled when the when option of the corresponding checkpointing environment containes an appropriate flag (see `checkpoint(5)`). Interactive jobs (see `qsh(1)`, `qrsh(1)`, `qtcsh(1)`) are not rescheduled.

  The default for *reschedule_unknown* id: `00:00:00`

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *stat_log_time* – Sun Grid Engine software periodically logs a snapshot of the status of the queues currently configured in the cluster to disk. The time interval in seconds between consecutive snapshots is defined by *stat_log_time*.

  Changing *stat_log_time* takes immediate effect. The default for *stat_log_time* is 2 hours 30 minutes.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *max_unheard* – If `sge_qmaster(8)` could not contact or was not contacted by the execution daemon of a host for *max_unheard* seconds, all queues residing on that particular host are set to status unknown. `sge_qmaster(8)`, at least, should be contacted by the execution daemons in order to get the load reports. Thus, *max_unheard* should by greater than the *load_report_time* (see above).

  Changing *max_unheard* takes immediate effect. The default for *max_unheard* is 2 minutes 30 seconds.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *loglevel* – This parameter specifies the level of detail that Sun Grid Engine components such as `sge_qmaster(8)` or `sge_execd(8)` use to produce informative, warning or error messages which are logged to the `messages` files in the master and execution daemon spool directories (see the description of the *qmaster_spool_dir* and the *execd_spool_dir* parameter above). The following message levels are available:

  - `log_err` – All error events being recognized are logged.
  - `log_warning` – All error events being recognized and all detected signs of potentially erroneous behavior are logged.
  - `log_info` – All error events being recognized, all detected signs of potentially erroneous behavior and a variety of informative messages are logged.

    Changing *loglevel* will take immediate effect.

    The default for *loglevel* is `log_info`.

    This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *max_aj_instances* – This parameter defines the maximum amount of array task to be scheduled to run simultaneously per array job. An instance of an array task will be created within the master daemeon when it gets a start order from the scheduler. The instance will be destroyed when the array task finishes. Thus the parameter provides control mainly over the memory consumption of array jobs in the master and scheduler daemon. It is most useful for very large clusters and very large array jobs. The default for this parameter is `2000`. The value `0` will deactivate this limit and will allow the scheduler to start as many array job tasks as suitable resources are available in the cluster.

  Changing *max_aj_instances* will take immediate effect.This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *max_aj_tasks* – his parameter defines the maximum number of array job tasks within an array job. sge_qmaster(8) will reject all array job submissions which request more than max_aj_tasks array job tasks. The default for this parameter is 75000. The value 0 will deactivate this limit.

  Changing max_aj_tasks will take immediate effect.

This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *max_u_jobs* – The number of active (not finished) jobs which each Sun Grid Engine user can have in the system simultaneously is controlled by this parameter. A value greater than `0` defines the limit. The default value `0` means "unlimited." If the *max_u_jobs* limit is exceeded by a job submission, then the submission command exits with exit status `25` and an appropriate error message.

  Changing *max_u_jobs* will take immediate effect.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *max_jobs* – This parameter controls the number of active (not finished) jobs simultaneously allowed in a Sun Grid Engine system. A value greater than `0` defines the limit. The default value `0` means "unlimited." If the *max_jobs* limit is exceeded by a job submission, then the submission command exits with exit status `25` and an appropriate error message.

  Changing *max_jobs* will take immediate effect.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *enforce_project* – This parameter is only available for Sun Grid Engine, Enterprise Edition systems. It is not present in Sun Grid Engine.

  If set to `true`, users are required to request a project whenever submitting a job. See the `–P` option to `qsub(1)` for details.

  Changing `enforce_project` will take immediate effect. The default for `enforce_project` is `false`.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *enforce_user* – This parameter is only available for Sun Grid Engine, Enterprise Edition systems. It is not present in Sun Grid Engine.

  If set to `true`, a Sun Grid Engine, Enterprise Edition `user(5)` must exist to allow for job submission. Jobs are rejected if no corresponding user exists.

  Changing `enforce_user` will take immediate effect. The default for `enforce_user` is `false`.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *set_token_cmd* – This parameter is only present if your Sun Grid Engine system is licensed to support AFS.

Set_token_cmd points to a command which sets and extends AFS tokens for Sun Grid Engine jobs. In the standard Sun Grid Engine AFS distribution, it is supplied as a script which expects two command line parameters. It reads the token from `STDIN`, extends the token's expiration time and sets the token:

```
<set_token_cmd> <user> <token_extend_after_seconds>
```

As a shell script this command will call the programs:

- `SetToken`
- `forge`

which are provided by your distributor as source code. The script looks as follows:

```
#!/bin/sh

# set_token_cmd

forge -u $1 -t $2 | SetToken
```

Since it is necessary for `forge` to read the secret AFS server key, a site might wish to replace the *set_token_cmd* script by a command, which connects to a self written daemon at the AFS server. The token must be forged at the AFS server and returned to the local machine, where `SetToken` is executed.

Changing *set_token_cmd* will take immediate effect. The default for *set_token_cmd* is none.

The global configuration entry for this value may be overwritten by the execution host local configuration.

- *pag_cmd* – This parameter is only present if your Sun Grid Engine system is licensed to support AFS.

  The path to your `pagsh` is specified via this parameter. The `sge_shepherd(8)` process and the job run in a `pagsh`. Ask your AFS administrator for details.

  Changing *pag_cmd* will take immediate effect. The default for *pag_cmd* is none.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *token_extend_time* – This parameter is only present if your Sun Grid Engine system is licensed to support AFS.

  This parameter sets the time period for which AFS tokens are periodically extended. Sun Grid Engine will call the token extension 30 minutes before the tokens expire until jobs have finished and the corresponding tokens are no longer required.

Changing *token_extend_time* will take immediate effect. The default for *token_extend_time* is 24:0:0, i.e. 24 hours.

The global configuration entry for this value may be overwritten by the execution host local configuration.

- *gid_range* – This parameter is only available for Sun Grid Engine, Enterprise Edition systems. It is not present in Sun Grid Engine.

  The *gid_range* is a comma-separated list of range expressions of the form *n-m* (*n* as well as *m* being positive non-zero integer numbers), where *m* is an abbreviation for *m-m*. These numbers are used in `sge_execd(8)` to identify processes belonging to the same job.

  Each `sge_execd(8)` may use a separate set of group ids for this purpose. All numbers in the group id range have to be unused supplementary group ids on the system, where the `sge_execd(8)` is started.

  Changing *gid_range* will take immediate effect. There is no default for *gid_range*. The administrator will have to assign a value for *gid_range* during installation of Sun Grid Engine, Enterprise Edition.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *qmaster_params* – A list of additional parameters can be passed to the Sun Grid Engine `qmaster`. The following values are recognized:

  - `ENABLE_FORCED_QDEL` – If this parameter is set, non-administrative users can foce deletion of their own jobs via the `-f` option of `qdel(1)`. Without this parameter, forced deletion of jobs is only allowed by the Sun Grid Engine manager or operator.

Note – Forced deletion for jobs is executed differently depending on whether users are Sun Grid Engine administrators or not. In case of administrative users, the jobs are removed from the internal database of Sun Grid Engine immediately. For regular users, the equivalent of a normal `qdel(1)` is executed first, and deletion is forced only if the normal cancellation was unsuccessful.

  Changing *qmaster_params* will take immediate effect. The default for *qmaster_params* is `none`.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

  - `FORBID_RESCHEDULE` – If this parameter is set, re-queuing of jobs cannot be initiated by the job script which is under control of the user. Without this parameter jobs returning the value 99 are rescheduled. This can be used to cause the job to be restarted at a different machine, for instance if there are not enough resources on the current one.

  - DISABLE_AUTO_RESCHEDULING – If set to true or 1, the *reschedule_unknown* parameter is not taken into account.

  Changing *qmaster_params* will take immediate effect. The default for *qmaster_params* is none.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *schedd_params* – This parameter is foreseen for passing additional parameters to the Sun Grid Engine scheduler. The following values are recognized currently:
  - FLUSH_SUBMIT_SEC, FLUSH_FINISH_SEC – The parameters are provided for tuning the system´s scheduling behavior. By default, a scheduler run is triggered in the scheduler interval which is defined in the scheduler configuration sched_conf(5), parameter *schedule_interval*.

    The parameters FLUSH_SUBMIT_SEC/FLUSH_FINISH_SEC define the time gaps between triggering a scheduler run and the submitting/finishing of a job.

    The most immediate scheduler reaction can be activated by setting both values to 0. The default scheduling behavior is enforced by either removing the parameters or setting them to a value of -1.
  - CLASSIC_SGEEE_SCHEDULING – If set to true or 1, use the original Sun Grid Engine, Enterprise Edition pending job scheduling algorithm. The original Sun Grid Engine, Enterprise Edition pending job scheduling algorithm calculates the tickets for pending jobs by dividing up the tickets in each scheduling policy among all the active and pending jobs. The pending job list is then ordered based on the tickets assigned to the pending jobs. The default value is false. This parameter is only valid in a Sun Grid Engine, Enterprise Edition system.
  - POLICY_HIERARCHY – This parameter sets up a dependency chain of policies. Each policy in the dependency chain is influenced by the previous policies and influences the following policies. A typical scenario is to assign precedence for the override policy over the share-based policy. The override policy determines in such a case how share-based tickets are assigned among jobs of the same user or project. Note that all policies contribute to the ticket amount assigned to a particular job regardless of the policy hierarchy definition. Yet the tickets calculated in each of the policies can be different, depending on POLICY_HIERARCHY.

    The POLICY_HIERARCHY parameter can be a up to four letter combination of the first letters of the four policies S(hare-based), F(unctional), D(eadline), and O(verride). So a value OFSD means that the override policy takes precedence over the functional policy, which influences the share-based policy and which finaly preceeds the deadline policy. Fewer than four letters means that some of the policies do not influence other policies and also are not influenced by other policies. So a value of FS means that the functional policy influences the share-based policy and that there is no interference with the other policies.

  The special value NONE switches off policy hierarchies.

  - SHARE_OVERRIDE_TICKETS – If set to true or 1, override tickets of any override object instance are shared equally among all jobs associated with the object. If set to false or 0, each job gets the full value of the override tickets associated with the object. The default value is true. This parameter is only valid in a Sun Grid Engine, Enterprise Edition system.
  - SHARE_DEADLINE_TICKETS – If set to true or 1, the total deadline tickets are shared among all deadline jobs. If set to false or 0, each deadline job will receive the total deadline tickets once the job deadline has been reached. The default value is true. This parameter is only valid in a Sun Grid Engine, Enterprise Edition system.
  - MAX_FUNCTIONAL_JOBS_TO_SCHEDULE – This is the maximum number of pending jobs to schedule in the functional policy. The default value is 200. This parameter is only valid in a Sun Grid Engine, Enterprise Edition system.
  - MAX_PENDING_TASKS_PER_JOB – The maximum number of subtasks per pending array job to schedule. This parameter exists in order to reduce scheduling overhead. The default value is 50. This parameter is only valid in a Sun Grid Engine, Enterprise Edition system.
  - PROFILE – If set, the scheduler logs profiling information, summarizing each scheduling run.

  Changing *schedd_params* will take immediate effect. The default for *schedd_params* is none.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *execd_params* – This parameter is foreseen for passing additional parameters to the Sun Grid Engine execution daemon. The following values are recognized:
  - ACCT_RESERVED_USAGE – If this parameter is set to true, for reserved usage is used for the accounting entries cpu, mem and io instead of the measured usage.
  - KEEP_ACTIVE – This value should only be set for debugging purposes. If set to true, the execution daemon will not remove the spool directory maintained by sge_shepherd(8) for a job.
  - NO_REPRIORITIZATION – If set to true or 1, jobs are not repriorized dynamically. Without setting this attribute, the priority parameter in sched_conf(5) has no durable effect in a Sun Grid Engine, Enterprise Edition system because of dynamic reprioritization performed by the PTF module inside sge_execd(8).
  - PTF_MIN_PRIORITY, PTF_MAX_PRIORITY – The parameters are only available in a Sun Grid Engine, Enterprise Edition system.

The maximum/minumum priority which Sun Grid Engine, Enterprise Edition will assign to a job. Typically this is a negative/postive value in the range of -20 (maximum) to 19 (minimum) for systems which allow setting of priorties with the `nice(2)` system call. Other systems may provide different ranges.

The default priority range (varies from system to system) is installed either by removing the parameters or by setting a value of `-999`.

See the `messages` file of the execution daemon for the predefined default value on your hosts. The values are logged during the startup of the execution daemon.

- `NOTIFY_KILL` – The parameter allows you to change the notification signal for the signal `SIGKILL` (see `-notify` option of `qsub(1)`). The parameter either accepts signal names (use the `-l` option of `kill(1)`) or the special value `none`. If set to `none`, no notification signal will be sent. If it is set to `TERM`, for instance, or another signal name, then this signal will be sent as notification signal.
- `NOTIFY_SUSP` – With this parameter it is possible to modify the notification signal for the signal SIGSTOP (see `-notify` parameter of `qsub(1)`). The parameter either accepts signal names (use the `-l` option of `kill(1)`) or the special value `none`. If set to `none`, no notification signal will be sent. If it is set to `TSTP`, for instance, or another signal name then this signal will be sent as notification signal.
- `SET_SGE_ENV`, `SET_COD_ENV`, `SET_GRD_ENV` – In `qsub(1)` you can find a list of environment variables that are exported into the execution environment of each Sun Grid Engine, Enterprise Edition job. Some of these variables names begin with the prefix `SGE_`. In the predecessor products of Sun Grid Engine, Enterprise Edition (Codine, GRD), the prefixes `COD_` and `GRD_` were used for those variables. This makes it necessary to update all job scripts relying one of those environment variables. The parameters, `SET_SGE_ENV`, `SET_COD_ENV` and `SET_GRD_ENV` can be used to accomplish a smoother transition from Codine/GRD to Sun Grid Engine, Enterprise Edition.

  Each of those parameters makes sure that a set of environment variables mentioned in `qsub(1)` and beginning with the corresponding prefix will be exported into the job environmnet. Every parameter can be set to `true`, `1` or `false`, `0`. Defaults are `SET_SGE_ENV=1`, `SET_COD_ENV=0` and `SET_GRD_ENV=0`.

  If all three parameters are set to `false` or `0`, jobs might fail as none of the corresponding environment variables would be set. Therefore, in this case, the environment variables beginning with the prefix `SGE_` are set.
- `SHARETREE_RESERVED_USAGE` – If this parameter is set to `true`, reserved usage is taken for the Sun Grid Engine, Enterprise Edition share tree consumption instead of measured usage.

  - USE_QSUB_GID – If this parameter is set to true, the primary group id being active when a job was submitted will be set to become the primary group id for job execution. If the parameter is not set, the primary group id as defined for the job owner in the execution host passwd(5) file is used.

    This feature is only available for jobs submitted via qsub(1), qrsh(1), qmake(1), and qtcsh(1). Also, it only works for qrsh(1) jobs (and thus also for qtcsh(1) and qmake(1)) if rsh and rshd components are used which are provided with Sun Grid Engine (i.e., the rsh_daemon and rsh_command parameters may not be changed from the default).

  Changing *execd_params* will take immediate effect. The default for *execd_params* is none.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *admin_user* – This parameter is the administrative user account used by Sun Grid Engine for all internal file handling (status spooling, message logging, etc.). Can be used in cases where the root account does not have the corresponding file access permissions (e.g., on a shared file system without global root read/write access).

  Because it is a parameter that is set at installation time, changing the *admin_user* parameter on a running system is not supported. Changing it on a shut-down cluster is possible, but if access to the Sun Grid Engine spooling area is interrupted, this will result in unpredictable behavior.

  The *admin_user* parameter has no default value, but instead it is defined during the master installation procedure.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *finished_jobs* – Sun Grid Engine software stores a certain number of *just finished* jobs to provide post mortem status information. The *finished_jobs* parameter defines the number of finished jobs being stored. If this maximum number is reached, the eldest finished job will be discarded for every now job being added to the finished job list.

  Changing *finished_jobs* will take immediate effect. The default for *finished_jobs* is 0.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *qlogin_daemon* – This parameter specifies the executable that is to be started on the server side of a qlogin(1) request. Usually this is the fully qualified path name of the system's telnet daemon. If no value is given, a specialized Sun Grid Engine component is used.

  Changing *qlogin_daemon* will take immediate effect. The default for *qlogin_daemon* is none.

The global configuration entry for this value may be overwritten by the execution host local configuration.

- *qlogin_command* – This is the command to be executed on the client side of a `qlogin(1)` request. Usually this is the fully qualified path name of the systems's `telnet` client program. If no value is given, a specialized Sun Grid Engine component is used. It is automatically started with the target host and port number as parameters.

  Changing *qlogin_command* will take immediate effect. The default for *qlogin_command* is `none`.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *rlogin_daemon* – This parameter specifies the executable that is to be started on the server side of a `qrsh(1)` request without a command argument to be executed remotely. Usually this is the fully qualified path name of the system's `rlogin` daemon. If no value is given, a specialized Sun Grid Engine component is used.

  Changing *rlogin_daemon* will take immediate effect. The default for *rlogin_daemon* is `none`.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *rlogin_command* – This is the command to be executed on the client side of a `qrsh(1)` request without a command argument to be executed remotely. Usually this is the fully qualified path name of the systems's `rlogin` client program. If no value is given, a specialized Sun Grid Engine component is used. The command is automatically started with the target host and port number as parameters like required for `telnet(1)`. The Sun Grid Engine `rlogin` client has been extended to accept and use the port number argument. You can only use clients, such as `ssh`, which also understand this syntax.

  Changing *rlogin_command* will take immediate effect. The default for *rlogin_command* is `none`.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *rsh_daemon* – This parameter specifies the executable that is to be started on the server side of a `qrsh(1)` request with a command argument to be executed remotely. Usually this is the fully qualified path name of the system's `rsh` daemon. If no value is given, a specialized Sun Grid Engine component is used.

  Changing *rsh_daemon* will take immediate effect. The default for *rsh_daemon* is `none`.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *rsh_command* – This is the command to be executed on the client side of a `qrsh(1)` request with a command argument to be executed remotely. Usually this is the fully qualified path name of the systems's `rsh` client program. If no value is given, a specialized Sun Grid Engine component is used. The command is automatically started with the target host and port number as parameters like required for `telnet(1)`plus the command with its arguments to be executed remotely. The Sun Grid Engine `rsh` client has been extended to accept and use the port number argument. You can only use clients, such as `ssh`, which also understand this syntax.

  Changing *rsh_command* will take immediate effect. The default for *rsh_command* is `none`.

  The global configuration entry for this value may be overwritten by the execution host local configuration.

- *ignore_fqdn* – This parameter causes the system to ignore the fully qualified domain name component of host names. The parameter should be set if all hosts belonging to a Sun Grid Engine cluster are part of a single DNS domain. It is switched on if set to either `true` or `1`. Switching it on may solve problems with load reports due to different host name resolutions across the cluster.

  Because it is a parameter that is set at installation time, changing the *ignore_fqdn* parameter on a running system is not supported. The default for *ignore_fqdn* is `true`.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

- *default_domain* – This parameter is needed only if your Sun Grid Engine cluster covers hosts belonging to more than a single DNS domain. In this case it can be used if your host name-resolving yields both qualified and unqualified host names for the hosts in one of the DNS domains. The value of *default_domain* is appended to the unqualified host name to define a fully-qualified host name. The *default_domain* parameter will have no effect if *ignore_fqdn* is set to `true`.

  Because it is a parameter that is set at installation time, changing the *default_domain* parameter on a running system is not supported. The default for *default_domain* is `none`, in which case it will not be used.

  This value is a global configuration parameter only. It cannot be overwritten by the execution host local configuration.

See Also

`sge_intro(1)`, `csh(1)`, `qconf(1)`, `qsub(1)`, `rsh(1)`, `sh(1)`, `getpwnam(3)`, `queue_conf(5)`, `sched_conf(5)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_shepherd(8)`, `cron(8)`, *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide, Sun Grid Engine 5.3 Administration and User's Guide*

Copyright

See sge_intro(1) for a full statement of rights and permissions.

sge_h_aliases(5)

Name

sge_h_aliases – Sun Grid Engine host aliases file format

Description

All Sun Grid Engine components use a host name resolving service provided by sge_commd(8) to identify hosts via a unique host name. sge_commd(8) itself references standard UNIX directory services such as DNS, NIS and /etc/hosts to resolve host names. In rare cases, these standard services cannot be set up cleanly and Sun Grid Engine communication daemons running on different hosts are unable to automatically determine a unique host name for one or all hosts which can be used on all hosts. In such situations, a Sun Grid Engine host aliases file can be used to provide the communication daemons with a private and consistent host name resolution database.

The default location for the host aliases file is:
<*sge_root*>/<*cell*>/common/host_aliases

To use a different host aliases file, it must be specified explicitly to sge_commd(8) via the -a command line option.

Format

For each host a single line must be provided with a blank, comma or semicolon separated list of hostname aliases. The first alias is defined to be the unique host name which will be used by all Sun Grid Engine components using the hostname aliasing service of the sge_commd(8).

See Also

`sge_intro(1)`, `sge_commd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_pe(5)

Name

`sge_pe` – Sun Grid Engine parallel environment configuration file format

Description

Parallel environments are parallel programming and runtime environments allowing for the execution of shared memory or distributed memory parallelized applications. Parallel environments usually require some kind of setup to be operational before starting parallel applications. Examples for common parallel environments are shared memory parallel operating systems and the distributed memory environments Parallel Virtual Machine (PVM) or Message Passing Interface (MPI).

`sge_pe` allows for the definition of interfaces to arbitrary parallel environments. Once a parallel environment is defined or modified with the `-ap` or `-mp` options to `qconf(1)` the environment can be requested for a job via the `-pe` switch to `qsub(1)` together with a request of a range for the number of parallel process to be allocated by the job. Additional `-l` options may be used to specify the job requirement to further detail.

Format

The format of a `sge_pe` file is defined as follows:

- *pe_name* – This is the name of the parallel environment. To be used in the `qsub(1)` `-pe` switch.

- *queue_list* – This is a comma-separated list of queues to which parallel jobs belonging to this parallel environment have access to.
- *slots* – This is the number of parallel processes being allowed to run in total under the parallel environment concurrently.
- *user_lists* – This is a comma-separated list of user access list names (see `access_list(5)`). Each user contained in at least one of the enlisted access lists has access to the parallel environment. If the *user_lists* parameter is set to NONE (the default) any user has access being not explicitly excluded via the *xuser_lists* parameter described below. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to the parallel environment.
- *xuser_lists* – The *xuser_lists* parameter contains a comma separated list of so called user access lists as described in `access_list(5)`. Each user contained in at least one of the enlisted access lists is not allowed to access the parallel environment. If the *xuser_lists* parameter is set to NONE (the default) any user has access. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to the parallel environment.
- *start_proc_args* – This is the invocation command line of a startup procedure for the parallel environment. The startup procedure is invoked by `sge_shepherd(8)` prior to executing the job script. Its purpose is to setup the parallel environment correspondingly to its needs. An optional prefix "user@" specifies the user under which this procedure is to be started. The standard output of the startup procedure is redirected to the file *REQName.poJID* in the job's working directory (see `qsub(1)`), with *REQName* being the name of the job as displayed by `qstat(1)` and *JID* being the job's identification number. Likewise, the standard error output is redirected to *REQName*.pe*JID*

  The following special variables being expanded at runtime can be used (besides any other strings which have to be interpreted by the start and stop procedures) to constitute a command line:

  - *$pe_hostfile* – This is the path name of a file containing a detailed description of the layout of the parallel environment to be setup by the startup procedure. Each line of the file refers to a host on which parallel processes are to be run. The first entry of each line denotes the hostname, the second entry the number of parallel processes to be run on the host and the third entry a processor range to be used in case of a multiprocessor machines.
  - *$host* – This is the name of the host on which the startup or stop procedures are started.
  - *$job_owner* – This is the user name of the job owner.
  - *$job_id* – This is Sun Grid Engine software's unique job identification number.
  - *$job_name* – This is the name of the job.
  - *$pe* – This is the name of the parallel environment in use.
  - *$pe_slots* – This is the number of slots granted for the job.

  - *$processors* – The *processors* string as contained in the queue configuration (see `queue_conf(5)`) of the master queue (the queue in which the startup and stop procedures are started).
  - *$queue* – This is the master queue, i.e. the queue in which the startup and stop procedures are started.

- *stop_proc_args* – This is the invocation command line of a shutdown procedure for the parallel environment. The shutdown procedure is invoked by `sge_shepherd(8)` after the job script has finished. Its purpose is to stop the parallel environment and to remove it from all participating systems. An optional prefix "user@" specifies the user under which this procedure is to be started. The standard output of the stop procedure is also redirected to the file *REQName*.po*JID* in the job's working directory (see `qsub(1)`), with *REQName* being the name of the job as displayed by `qstat(1)` and *JID* being the job's identification number. Likewise, the standard error output is redirected to *REQName*.pe*JID*

  The same special variables as for *start_proc_args* can be used to constitute a command line.

- *signal_proc_args* – This is the invocation command line of a signalling procedure for the parallel environment. The signalling procedure is invoked by `sge_shepherd(8)` after whenever a signal is sent to the parallel job via `qmod(1)`, `qdel(1)` or in case of a migration request. Its purpose is to signal all components of the parallel environment and their associated application processes correspondingly. The standard output of the signalling procedure is also redirected to the file *REQName*.po*JID* in the job's working directory (see `qsub(1)`), with *REQName* being the name of the job as displayed by *qstat(1)* and *JID* being the job's identification number. Likewise, the standard error output is redirected to *REQName*.pe*JID*

  The same special variables as for *start_proc_args* can be used to constitute a command line.

- *allocation_rule* – The allocation rule is interpreted by `sge_schedd(8)` and helps the scheduler to decide how to distribute parallel processes among the available machines. If, for instance, a parallel environment is built for shared memory applications only, all parallel processes have to be assigned to a single machine, no matter how much suitable machines are available. If, however, the parallel environment follows the distributed memory paradigm, an even distribution of processes among machines may be favorable.

  The current version of the scheduler only understands the following allocation rules:

  - <*int*> – This is an integer number fixing the number of processes per host. If the number is 1, all processes have to reside on different hosts. If the special denominator `$pe_slots` is used, the full range of processes as specified with the `qsub(1) -pe` switch has to be allocated on a single host (no matter which value belonging to the range is finally chosen for the job to be allocated).

  - `$fill_up`– Starting from the best suitable host/queue, all available slots are allocated. Further hosts and queues are "filled up" as long as a job still requires slots for parallel tasks.
  - `$round_robin`– From all suitable hosts a single slot is allocated until all tasks requested by the parallel job are dispatched. If more tasks are requested than suitable hosts are found, allocation starts again from the first host. The allocation scheme walks through suitable hosts in a best-suitable-first order.
- *control_slaves* – This parameter can be set to TRUE or FALSE (the default). It indicates whether Sun Grid Engine is the creator of the slave tasks of a parallel application via `sge_execd(8)` and `sge_shepherd(8)` and thus has full control over all processes in a parallel application, which enables capabilities such as resource limitation and correct accounting. However, to gain control over the slave tasks of a parallel application, a sophisticated PE interface is required, which works closely together with Sun Grid Engine facilities. Such PE interfaces are available through your local Sun Grid Engine support office.

  Please set the control_slaves parameter to false for all other PE interfaces.
- *job_is_first_task* – This parameter is only checked if `control_slaves` (see above) is set to TRUE and thus Sun Grid Engine is the creator of the slave tasks of a parallel application via `sge_execd(8)` and `sge_shepherd(8)`. In this case, a sophisticated PE interface is required closely coupling the parallel environment and Sun Grid Engine. The documentation accompanying such PE interfaces will recommend the setting for `job_is_first_task`.

  The `job_is_first_task` parameter can be set to TRUE or FALSE. A value of TRUE indicates that the Sun Grid Engine job script already contains one of the tasks of the parallel application, while a value of FALSE indicates that the job script (and its child processes) is not part of the parallel program.

Restrictions

Note – The functionality of the startup, shutdown and signalling procedures remains the full responsibility of the administrator configuring the parallel environment. Sun Grid Engine will just invoke these procedures and evaluate their exit status. If the procedures do not perform their tasks properly or if the parallel environment or the parallel application behave unexpectedly, Sun Grid Engine has no means to detect this.

See Also

`sge_intro(1)`, `qconf(1)`, `qdel(1)`, `qmod(1)`, `qsub(1)`, `access_list(5)`, `sge_qmaster(8)`, `sge_schedd(8)`, `sge_shepherd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

complex(5)

Name

`complex` – Sun Grid Engine complexes configuration file format

Description

Complex reflects the format of the Sun Grid Engine complexes configuration. The definition of complexes provides all pertinent information concerning the resource attributes a user may request for a Sun Grid Engine job via the `qsub(1) -l` option and for the interpretation of these parameters within the Sun Grid Engine system.

The complexes configuration files should not be accessed directly. In order to add or modify complexes, the `qconf(1)` options `-Ac`, `-ac`, `-Mc` and `-mc` should be used instead. While the `-Ac` and `-Mc` options take a *complex* configuration file as an argument, the `-ac` and `-mc` options bring up an editor filled in with a template *complex* configuration or the configuration of an existing complex.

The Sun Grid Engine complexes object integrates four types of complexes:

- **The Queue Complex** – This is referenced by the special name, *queue*.

  In its default form it contains a selection of parameters in the queue configuration as defined in *queue_conf(5)*. The queue configuration parameters being requestable for a job by the user in principal are:

  - *qname*
  - *hostname*
  - *notify*

- *calendar*
- *min_cpu_interval*
- *tmpdir*
- *seq_no*
- *s_rt*
- *h_rt*
- *s_cpu*
- *h_cpu*
- *s_data*
- *h_data*
- *s_stack*
- *h_stack*
- *s_core*
- *h_core*
- *s_rss*
- *h_rss*

The queue complex can be extended if further attributes are intended to be available for each queue. The queue complex defines the characteristics (such as the data type) of the attributes it contains. A value setting for the queue complex attributes is defined by the queue configuration for each queue in case of the standard parameters enlisted above, or by the *complex_values* entry in the queue configuration (see `queue_conf(5)` for details) if a parameter has been added to the default queue complex. If no definition for the value in the *complex_values* entry of the queue configuration is given in the latter case, the value is set as defined by the `value` field described below.

- **The Host Complex** – Referenced by the special name, *host*, this complex contains the characteristics definition of all attributes which are intended to be managed on a host basis. The standard set of host related attributes consists of two categories, but it may be enhanced like the queue complex as described above. The first category is built by several queue configuration attributes which are particularly suitable to be managed on a host basis. These attributes are:
  - *slots*
  - *s_vmem*
  - *h_vmem*
  - *s_fsize*
  - *h_fsize*

  (Refer to `queue_conf(5)` for details).

Note – Defining these attributes in the host complex is no contradiction to having them also in the queue configuration. It allows maintaining the corresponding resources on a host level and at the same time on a queue level. Total virtual free memory (h_vmem) can be managed for a host, for example, and a subset of the total amount can be associated with a queue on that host.

The second attribute category in the standard host complex are the default load values Every `sge_execd(8)` periodically reports load to `sge_qmaster(8)`. The reported load values are either the standard Sun Grid Engine load values such as the CPU load average (see `uptime(1)`) or load values defined by the Sun Grid Engine administration (see the `load_sensor` parameter in the cluster configuration `sge_conf(5)`, the *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide* and the *Sun Grid Engine 5.3 Administration and User's Guide* for details). The characteristics definition for the standard load values is part of the default host complex, while administrator defined load values require extension of the host complex. Refer to the file, <*sge_root*>`/doc/load_parameters.asc` for detailed information on the standard set of load values.

The host complex commonly is not only extended to include non-standard load parameters, but also to manage host-related resources such as the number of software licenses being assigned to a host or the available disk space on a host local file system.

A concrete value for a particular host complex attribute is determined by either an associated queue configuration in the case of the queue configuration derived attributes, a reported load value or the explicit definition of a value in the `complex_values` entry of the corresponding host configuration (see `host_conf(5)`). If none of the above is available (e.g. the value is supposed to be a load parameter, but `sge_execd(8)` does not report a load value for it), the `value` field described below is used.

- **The Global Complex** – This is referenced by the special name, *global*.

  The entries configured in the global complex refer to cluster wide resource attributes, such as the number of available "floating" licenses of a particular software or the free disk space on a network wide available filesystem. Global resource attributes can also be associated with load reports, if the corresponding load report contains the `GLOBAL` identifier (see the corresponding section in the *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide* or the *Sun Grid Engine 5.3 Administration and User's Guide* for details). Global load values can be reported from any host in the cluster. There are no global load values reported by Sun Grid Engine by default and hence there is no default global complex configuration.

  Concrete values for global complex attributes are either determined by global load reports or by explicit definition in the `complex_values` parameter of the "global" host configuration (see `host_conf(5)`). If none of both is present (e.g. a load value has not yet been reported) the `value` field described below is used.

- **User Defined Complexes** – By setting up user defined complexes, the Sun Grid Engine administrator has the ability to extend the set of attributes managed by Sun Grid Engine while restricting the influence of those attributes to particular queues and/or hosts. A user complex is just a named collection of attributes and the corresponding definition as to how these attributes are to be handled by Sun Grid Engine software. One or more of these user defined complexes can be attached to a queue and/or host via the `complex_list` queue and host configuration parameter (see `queue_conf(5)` and `host_conf(5)`). The attributes defined in all assigned complexes become available to the queue and the host respectively in addition to the default complex attributes.

Concrete values for user defined complexes have to be set by the `complex_values` parameter in the queue and host configuration or otherwise the `value` field described below is used.

Format

The principal format of a *complex* configuration is that of a tabulated list. Each line starting with a '#' character is a comment line. Each line despite comment lines define one element of the complex. An element definition line consists of the following 6 column entries per line (in the order of appearance):

- *name* – This is the name of the complex element to be used to request this attribute for a job in the `qsub(1) -l` option. An attribute *name* may appear only once across all complexes, i.e. the complex attribute definition is unique.
- *shortcut* – A shortcut for *name* which may also be used to request this attribute for a job in the `qsub(1) -l` option. An attribute *shortcut* may appear only once across all complexes, so as to avoid the possibility of ambiguous complex attribute references.
- *type* – This setting determines how the corresponding values are to be treated by Sun Grid Engine software internally in case of comparisons or in case of load scaling for the load complex entries:
  - With `INT` only raw integers are allowed.
  - With `DOUBLE` floating point numbers in double precision (decimal and scientific notation) can be specified.
  - With `TIME` time specifiers are allowed. Refer to `queue_conf(5)` for a format description.
  - With `MEMORY` memory size specifiers are allowed. Refer to `queue_conf(5)` for a format description.
  - With `BOOL` the strings TRUE and FALSE are allowed. When used in a load formula (refer to `sched_conf(5)`) TRUE and FALSE get mapped into '1' and '0'.
  - With `STRING` all strings are allowed and `strcmp(3)` is used for comparisons.
  - `CSTRING` is like `STRING` except comparisons are case insensitive.
  - `HOST` is like `CSTRING` but the string must be a valid host name.

- *value* – The *value* field is a pre-defined value setting for an attribute, which only has an effect if it is not overwritten while attempting to determine a concrete value for the attribute with respect to a queue, a host or the Sun Grid Engine cluster. The value field can be overwritten by the following.
  - The queue configuration values of a referenced queue
  - Host-specific and cluster related load values
  - Explicit specification of a value via the complex_values parameter in the queue or host configuration (see `queue_conf(5)` and `host_conf(5)` for details

  If none of the above is applicable, value is set for the attribute.
- *relop* – This is the *relation operator.* The relation operator is used when the value requested by the user for this parameter is compared against the corresponding value configured for the considered queues. If the result of the comparison is false, the job cannot run in this queue. Possible relation operators are `==`, `<`, `>`, `<=` and `>=`. The only valid operator for string type attributes is `==`.
- *requestable* – The entry can be used in a `qsub(1)` resource request if this field is set to `y` or `yes`. If set to `n` or `no`, this entry cannot be used by a user in order to request a queue or a class of queues. If the entry is set to `forced` or `f`, the attribute has to be requested by a job or it is rejected.
- *consumable* – The *consumable* parameter can be set to either `yes` (`y` abbreviated) or `no` (`n`). It can be set to `yes` only for numeric attributes (`INT`, `MEMORY`, `TIME`—see *type* above). If set to `yes` the consumption of the corresponding resource can be managed by Sun Grid Engine internal bookkeeping. In this case Sun Grid Engine accounts for the consumption of this resource for all running jobs and ensures that jobs are only dispatched if the Sun Grid Engine internal bookkeeping indicates enough available consumable resources. Consumables are an efficient means to manage limited resources such a available memory, free space on a file system, network bandwidth or floating software licenses.

  Consumables can be combined with default or user defined load parameters (see `sge_conf(5)` and `host_conf(5)`); e.g., load values can be reported for consumable attributes or the consumable flag can be set for load attributes. The Sun Grid Engine consumable resource management takes both the load (measuring availability of the resource) and the internal bookkeeping into account in this case, and makes sure that neither of both exceeds a given limit.

  To enable consumable resource management the basic availability of a resource has to be defined. This can be done on a cluster global, per host and per queue basis while these categories may supersede each other in the given order (i.e. a host can restrict availability of a cluster resource and a queue can restrict host and cluster resources). The definition of resource availability is performed with the *complex_values* entry in `host_conf(5)` and `queue_conf(5)`. The *complex_values* definition of the “global” host specifies cluster global consumable settings. To each consumable complex attribute in a *complex_values* list a value is assigned which denotes the maximum available amount for that resource. The internal bookkeeping will subtract from this total the assumed resource consumption by all running jobs as expressed through the jobs' resource requests.

Note – Jobs can be forced to request a resource and thus to specify their assumed consumption via the 'force' value of the requestable parameter (see above).

Note – A default resource consumption value can be pre-defined by the administrator for consumable attributes not explicitly requested by the job (see the default parameter below). This is meaningful only if requesting the attribute is not enforced as explained above.

See the *Sun Grid Engine 5.3 Administration and User's Guide* or the *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide* for examples on the usage of the consumable resources facility.

- default – This is meaningful only for consumable complex attributes (see *consumable* parameter above). Sun Grid Engine assumes the resource amount denoted in the default parameter implicitly to be consumed by jobs being dispatched to a host or queue managing the consumable attribute. Jobs explicitly requesting the attribute via the `-l` option to `qsub(1)` override this default value.

See Also

`sge_intro(1)`, `qconf(1)`, `qsub(1)`, `uptime(1)`, `host_conf(5)`, `queue_conf(5)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_schedd(8)`, *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide*, *Sun Grid Engine 5.3 Administration and User's Guide*

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

host_conf(5)

Name

host_conf – Sun Grid Engine execution host configuration file format

Description

Host_conf reflects the format of the template file for the execution host configuration. Via the -ae and -me options of the qconf(1) command, you can add execution hosts and modify the configuration of any execution host in the cluster. Default execution host entries are added automatically as soon as a sge_execd(8) registers to sge_qmaster(8) for the very first time from a certain host. The qconf(1) -sel switch can be used to display a list of execution host being currently configured in your Sun Grid Engine system. Via the -se option you can print the execution host configuration of a specified host.

The special host name, global, can be used to define cluster global characteristics.

Format

The format of a host_conf file is defined as follows.

- *hostname* – This is the name of the execution host.
- *load_scaling* – This is a comma-separated list of scaling values to be applied to each or part of the load values being reported by the sge_execd(8) on the host and being defined in the cluster global "host" complex (see complex(5)). The load scaling factors are intended to level hardware or operating system specific differences between execution hosts. If, for example, the load average value (load_avg in the "host" complex; see also uptime(1)) of a multiprocessor machine is to be compared with a single processor machine the load as reported by the single CPU host needs to be weighted up against the multiprocessor load (given the same CPU hardware) to be comparable. The load scaling factors are integers being multiplied with the reported load quantities to constitute weighted load values. Thus, following the example given above, the load value of the single processor machine needs to be multiplied by the number of processors of the single processor machine to become comparable.

The syntax of a load factor specification is as follows: First the name of the load value (as defined in the "host" complex) is given and, separated by an equal sign, the load scaling value is provided. No blanks are allowed in between the load_scaling value string.

The parameter *load_scaling* is not meaningful for the definition of the "global" host.

- *complex_list* – This is the comma separated list of administrator defined complexes (see `complex(5)` for details) to be associated with the host. Only complex attributes contained in the enlisted complexes and those from the "global" and "host" complex, which are implicitly attached to each host, can be used in the *complex_values* list below. In case of the "global" host, the "host" complex is not attached and only "global" complex attributes are allowed per default in the *complex_values* list of the "global" host.

  The default value for this parameter is NONE, i.e. no administrator defined complexes are associated with the host.

- *complex_values* – complex_values defines quotas for resource attributes managed via this host. Each complex attribute is followed by an "=" sign and the value specification compliant with the complex attribute type (see `complex(5)`). Quota specifications are separated by commas. Only attributes as defined in *complex_list* (see above) may be used.

  The quotas are related to the resource consumption of all jobs on a host in the case of consumable resources (see `complex(5)` for details on consumable resources) or they are interpreted on a per job slot basis in the case of non-consumable resources. Consumable resource attributes are commonly used to manage free memory, free disk space or available floating software licenses while non-consumable attributes usually define distinctive characteristics like type of hardware installed.

  For consumable resource attributes an available resource amount is determined by subtracting the current resource consumption of all running jobs on the host from the quota in the *complex_values* list. Jobs can only be dispatched to a host if no resource requests exceed any corresponding resource availability obtained by this scheme. The quota definition in the *complex_values* list is automatically replaced by the current load value reported for this attribute, if load is monitored for this resource and if the reported load value is more stringent than the quota. This effectively avoids oversubscription of resources.

Note – Load values replacing the quota specifications may have become more stringent because they have been scaled (see load_scaling above) and/or load adjusted (see `sched_conf(5)`). The `-F` option of `qstat(1)` and the load display in the `qmon(1)` queue control dialog (activated by clicking on a queue icon while the "Shift" key is pressed) provide detailed information on the actual availability of consumable resources and on the origin of the values taken into account currently.

Note – The resource consumption of running jobs (used for the availability calculation) as well as the resource requests of the jobs waiting to be dispatched either may be derived from explicit user requests during job submission (see the `-l` option to `qsub(1)`) or from a "default" value configured for an attribute by the administrator (see `complex(5)`). The `-r` option to `qstat(1)` can be used for retrieving full detail on the actual resource requests of all jobs in the system.

For non-consumable resources Sun Grid Engine simply compares the job's attribute requests with the corresponding specification in *complex_values* taking the relation operator of the complex attribute definition into account (see `complex(5)`). If the result of the comparison is "true", the host is suitable for the job with respect to the particular attribute. For parallel jobs each job slot to be occupied by a parallel task is meant to provide the same resource attribute value.

Note – Only numeric complex attributes can be defined as consumable resources and hence non-numeric attributes are always handled on a per job slot basis.

The default value for this parameter is NONE, i.e. no administrator defined resource attribute quotas are associated with the host.

- *load_values* – This entry cannot be configured but is only displayed in case of a `qconf(1) -se` command. All load values are displayed as reported by the `sge_execd(8)` on the host. The load values are enlisted in a comma separated list. Each load value start with its name, followed by an equal sign and the reported value.
- *processors* – This entry cannot be configured but is only displayed in case of a `qconf(1) -se` command. Its value is the number of processors which has been detected by `sge_execd(8)` on the corresponding host.
- *usage_scaling* – This entry is only present in a Sun Grid Engine, Enterprise Edition system. It is not available in Sun Grid Engine.

  The format is equivalent to *load_scaling* (see above), the only valid attributes to be scaled however are `cpu` for CPU time consumption, `mem` for Memory consumption aggregated over the life-time of jobs and `io` for data transferred via any I/O devices. The default NONE means "no scaling", i.e. all scaling factors are 1.
- resource_capability_factor – This entry is only present in a Sun Grid Engine, Enterprise Edition system. It is not available in Sun Grid Engine.

  The resource capability factor is used by Sun Grid Engine, Enterprise Edition when assigning jobs to execution hosts. The resource capability factor tells Sun Grid Engine, Enterprise Edition how the resources (CPU, memory, I/O, etc.) of one execution host compare to the resources of other execution hosts. This helps to ensure that a job requiring a large percentage of resources (i.e. lots of tickets) gets placed on an execution host containing a large percentage of the available resources. The load situation on the execution hosts is taken into account in addition, to guarantee that the selected host is both powerful enough and lightly loaded.

For example, you might consider setting your resource capability factors for each execution host based on the number of CPUs, the speed of the CPUs, and the installed main memory:

#_of_CPUs * (MHz/200) + GB_of_memory

This would give an execution host with 32 200 MHz CPUs and 10 gigabytes of memory a resource capability factor of 42, while an execution host with 24 200 MHz CPUs and 40 gigabytes of memory would get a resource capability factor of 64, i.e. memory has a significant impact in this example.

Other factors that you might want to consider in setting the resource capability factor are: the following.

- Job mix – CPU or memory bound jobs
- CPU benchmarks – comparison by CPU vendor
- Mmegaflops (MFLOPS) – for number crunching
- I/O capabilities – disk/network speed
- Available disk space – at the execution host

The resource capability factor is stored as a floating point double value. The range of values used is not important. Sun Grid Engine, Enterprise Edition only looks at the relation between values of different hosts.

See Also

`sge_intro(1)`, `qconf(1)`, `uptime(1)`, `complex(5)`, `sge_execd(8)`, `sge_qmater(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

hostgroup(5)

Name

hostgroup - Host group entry file format

Description

A hostgroup entry is used to merge host names to groups. Each hostgroup entry file defines one group. A group is referenced by the sign @ as first character of the name. At this point of implementation, you can use hostgroups in the usermapping(5) configuration. Inside a group definition file, you can also reference other groups. These groups are called *subgroups*.

Each line in the hostgroup entry file specifies a host name or a group that belongs to this group.

You can display a list of currently configured hostgroup entries via the qconf(1) -shgrpl option. You can show the contents of each listed hostgroup entry via the -shgrp switch. The output follows the hostgroup format description. You can create or modify hostgroup entries via the -ahgrp, -mhgrp, and -dhgrp options to qconf(1).

Format

A hostgroup entry contains at least two parameters.

- *group_name* keyword – The *group_name* keyword defines the hostgroup name. The rest of the textline after the keyword *group_name* will be taken as hostgroup name value.
- *hostname* – The name of the host that is now member of the group specified with *group_name*. If the first character of the *hostname* is a @ sign, the name is used to reference a hostgroup(5), which is taken as subgroup of this group.

Example

The following is a typical hostgroup entry.

```
group_name bigMachines

@calculate

speedhost
```

The entry will define a new hostgroup called bigMachines. In this hostgroup is the host speedhost and all members of the hostgroup, calculate.

See Also

qconf(1), usermapping(5)

Copyright

See sge_intro(1) for a full statement of rights and permissions.

project(5)

Name

project – Sun Grid Engine, Enterprise Edition project entry file format

Description

The project object is only available in case of a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine has no project object.

Jobs can be submitted to projects in Sun Grid Engine, Enterprise Edition and a project can be assigned with a certain level of importance via the functional or the override policy. This level of importance is then inherited by the jobs executing under that project.

A list of currently configured projects can be displayed via the `qconf(1) -sprjl` option. The contents of each enlisted project definition can be shown via the `-sprj` switch. The output follows the `project` format description. New projects can be created and existing can be modified via the `-aprj`, `-mprj` and `-dprj` options to `qconf(1)`.

Format

A project definition contains the following parameters:

- `name` – This is the project name.
- `oticket` – This is the amount of override tickets currently assigned to the project.
- `fshare` – This is the current functional share of the project.
- `facl` – This is a list of user access lists (ACLs - see `access_list(5)`) referring to those users being allowed to submit jobs to the project.
- `fxacl` – This is a list of user access lists (ACLs - see `access_list(5)`) referring to those users being not allowed to submit jobs to the project.

See Also

`sge_intro(1)`, `qconf(1)`, `access_list(5)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

qtask(5)

Name

qtask – file format of the qtask file.

Description

A qtask file defines which commands are submitted to Sun Grid Engine for remote execution by qtcsh(1). The qtask file optionally may contain qrsh(1) command-line parameters. These parameters are passed to the qrsh(1) command being used by qtcsh to submit the commands.

A cluster global qtask file defining cluster wide defaults and a user specific qtask file eventually overriding and enhancing those definitions are supported. The cluster global file resides at *<sge_root>*/*<cell>*/common/qtask, while the user specific file can be found at ~/.qtask. An exclamation mark preceding command definitions in the cluster global can be used by the administrator to deny overriding of such commands by users.

Format

The principle format of the qtask file is that of a tabulated list. Each line starting with a '#' character is a comment line. Each line despite comment lines defines a command to be started remotely.

Definition starts with the command name that must match exactly the name as typed in a qtcsh(1) command-line. Pathnames are not allowed in qtask files. Hence absolute or relative pathnames in qtcsh(1) command-lines always lead to local execution even if the commands itself are the same as defined in the qtask files.

The command name can be followed by an arbitrary number of qrsh(1) option arguments which are passed on to qrsh(1) by qtcsh(1). An exclamation mark prefixing the command in the cluster global qtask file prevents overriding by the user supplied qtask file.

Example

```
netscape -l a=solaris64 -v DISPLAY=myhost:0
grep -l h=filesurfer
verilog -l veri_lic=1
```

The `qtask` file above designates the applications netscape, `grep` and *verilog* for interactive remote execution through Sun Grid Engine software. Netscape is requested to run only on `Solaris64` architectures with the *DISPLAY* environment variable set to `myhost:0`, `grep` only runs on the host named `filesurfer` and `verilog` requests availability of a verilog license in order to get executed remotely.

See Also

`sge_intro(1)`, `qtcsh(1)`, `qrsh(1)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

queue_conf(5)

Name

`queue_conf` – Sun Grid Engine queue configuration file format

Description

`queue_conf` reflects the format of the template file for the queue configuration. Via the `–aq` and `–mq` options of the `qconf(1)` command, you can add queues and modify the configuration of any queue in the cluster.

The `queue_conf` parameters take as values strings, integer decimal numbers or boolean, time and memory specifiers as well as comma separated lists. A time specifier either consists of a positive decimal, hexadecimal or octal integer constant, in which case the value is interpreted to be in seconds, or is built by 3 decimal integer numbers separated by colon signs where the first number counts the hours, the second the minutes and the third the seconds. If a number would be zero it can be left out but the separating colon must remain
(e.g. 1:0:1 = 1::1 means 1 hours and 1 second).

Memory specifiers are positive decimal, hexadecimal or octal integer constants which may be followed by a multiplier letter. Valid multiplier letters are k, K, m and M, where k means multiply the value by 1000, K multiply by 1024, m multiply by 1000*1000 and M multiplies by 1024*1024. If no multiplier is present, the value is just counted in bytes.

Format

The following list of `queue_conf` parameters specifies the `queue_conf` content:

- *qname* – This is the name of the queue on the node (type string; template default: template).
- *hostname* – This is the fully-qualified host name of the node (type string; template default: *host.dom.dom.dom*).
- `seq_no` – In conjunction with the hosts load situation, at a time this parameter specifies this queue's position in the scheduling order within the suitable queues for a job to be dispatched under consideration of the *queue_sort_method* (see `sched_conf(5)`).

  Regardless of the `queue_sort_method` setting, `qstat(1)` reports queue information in the order defined by the value of the `seq_no`. Set this parameter to a monotonically increasing sequence. The type is *number* and the default is `0`.
- *load_thresholds* – This is a list of load thresholds. Already if one of the thresholds is exceeded no further jobs will be scheduled to the queues on this node and `qmon(1)` will signal an overload condition for this node. Arbitrary load values being defined in the "host" and "global" complexes (see `complex(5)` for details) can be used.

  The syntax is that of a comma separated list with each list element consisting of the name of a load value, an equal sign and the threshold value being intended to trigger the overload situation (e.g. load.avg=175,users_logged_in=5).

Note – Load values as well as consumable resources may be scaled differently for different hosts if specified in the corresponding execution host definitions (refer to `host_conf(5)` for more information). Load thresholds are compared against the scaled load and consumable values.

- *suspend_thresholds* – This is a list of load thresholds with the same semantics as that of the *load_thresholds* parameter (see above) except that exceeding one of the denoted thresholds initiates suspension of one of multiple jobs in the queue. See the `nsuspend` parameter below for details on the number of jobs which are suspended.
- *nsuspend* – This is the number of jobs which are suspended/enabled per time interval if at least one of the load thresholds in the *suspend_thresholds* list is exceeded or if no *suspend_threshold* is violated anymore respectively. *nsuspend* jobs are suspended in each time interval until no *suspend_thresholds* are exceeded anymore or all jobs in the queue are suspended. Jobs are enabled in the corresponding way if the *suspend_thresholds* are no longer exceeded. The time interval in which the suspensions of the jobs occur is defined in *suspend_interval* below.
- *suspend_interval* – This is the time interval in which further *nsuspend* jobs are suspended if one of the *suspend_thresholds* (see above for both) is exceeded by the current load on the host on which the queue is located. The time interval is also used when enabling the jobs.
- *priority* – The *priority* parameter specifies the `nice(2)` value at which jobs in this queue will be run. The type is *number* and the default is zero (which means no `nice` value is set explicitly). Negative values (up to -20) correspond to a higher scheduling priority. Positive values (up to +20) correspond to a lower scheduling priority. In a Sun Grid Engine, Enterprise Edition system, the value of *priority* has no effect because the Sun Grid Engine, Enterprise Edition system adjusts priorities dynamically to implement the Sun Grid Engine, Enterprise Edition entitlement policy goals. Dynamic priority adjustment can be switched off, however, by setting the *execd_param* `NO_REPRIORITIZATION` to `true` in the global or execution daemon local cluster configuration (see `sge_conf(5)`). In this case, the setting of *priority* also will have an effect in the Sun Grid Engine, Enterprise Edition system.
- *min_cpu_interval* – This is the time between two automatic checkpoints in case of transparently checkpointing jobs. The maximum of the time requested by the user via `qsub(1)` and the time defined by the queue configuration is used as checkpoint interval. Since checkpoint files may be considerably large and thus writing them to the file system may become expensive, users and administrators are advised to choose sufficiently large time intervals. *min_cpu_interval* is of type time and the default is 5 minutes (which usually is suitable for test purposes only).

- *processors* – This is a set of processors in case of a multiprocessor execution host can be defined to which the jobs executing in this queue are bound. The value type of this parameter is a range description like that of the `-pe` option of `qsub(1)` (e.g., 1-4,8,10) denoting the processor numbers for the processor group to be used. Obviously the interpretation of these values relies on operating system specifics and is thus performed inside `sge_execd(8)` running on the queue host. Therefore, the parsing of the parameter has to be provided by the execution daemon and the parameter is only passed through `sge_qmaster(8)` as a string.

  Currently, support is only provided for SGI multiprocessor machines running IRIX 6.2 and Digital UNIX multiprocessor machines. In the case of Digital UNIX only one job per processor set is allowed to execute at the same time; i.e. *slots* (see above) should be set to 1 for this queue.
- *qtype* – This is the type of queue. Currently one of *batch, interactive, parallel* or *checkpointing* or any combination in a comma separated list. The type is *string*; the default is `batch interactive parallel`.
- *rerun* – This defines a default behavior for jobs which are aborted by system crashes or manual "violent" (via `kill(1)`) shutdown of the complete Sun Grid Engine system (including the `sge_shepherd(8)` of the jobs and their process hierarchy) on the queue host. As soon as `sge_execd(8)` is restarted and detects that a job has been aborted for such reasons it can be restarted if the jobs are restartable. A job may not be restartable, for example, if it updates databases (first reads then writes to the same record of a database/file) because the abortion of the job may have left the database in an inconsistent state. If the owner of a job wants to overrule the default behavior for the jobs in the queue the `-r` option of `qsub(1)` can be used.

  The type of this parameter is boolean, thus either TRUE or FALSE can be specified. The default is FALSE, i.e. do not restart jobs automatically.
- *slots* – This is the maximum number of concurrently executing jobs allowed in the queue. Type is number.
- *tmpdir* – The *tmpdir* parameter specifies the absolute path to the base of the temporary directory filesystem. When `sge_execd(8)` launches a job, it creates a uniquely-named directory in this filesystem for the purpose of holding scratch files during job execution. At job completion, this directory and its contents are removed automatically. The environment variables `TMPDIR` and `TMP` are set to the path of each jobs scratch directory (type string; default: `/tmp`).
- *shell* – If either *posix_compliant* or *script_from_stdin* is specified as the *shell_start_mode* parameter in `sge_conf(5)` the *shell* parameter specifies the executable path of the command interpreter (e.g. `sh(1)` or `csh(1)`) to be used to process the job scripts executed in the queue. The definition of *shell* can be overruled by the job owner via the `qsub(1) -S` option.

  The type of the parameter is string. The default is `/bin/csh`.

- *shell_start_mode* – This parameter defines the mechanisms which are used to actually invoke the job scripts on the execution hosts. The following values are recognized:
  - *unix_behavior* – If a user starts a job shell script under UNIX interactively by invoking it just with the script name the operating system's executable loader uses the information provided in a comment such as `#!/bin/csh` in the first line of the script to detect which command interpreter to start to interpret the script. This mechanism is used by Sun Grid Engine when starting jobs if *unix_behavior* is defined as *shell_start_mode*.
  - *posix_compliant* – POSIX does not consider first script line comments such a `#!/bin/csh` as being significant. The POSIX standard for batch queuing systems (P1003.2d) therefore requires a compliant queuing system to ignore such lines but to use user specified or configured default command interpreters instead. Thus, if *shell_start_mode* is set to *posix_compliant* Sun Grid Engine will either use the command interpreter indicated by the `-S` option of the `qsub(1)` command or the *shell* parameter of the queue to be used (see above).
  - *script_from_stdin* – Setting the *shell_start_mode* parameter either to *posix_compliant* or *unix_behavior* requires you to set the umask in use for `sge_execd(8)` such that every user has read access to the active_jobs directory in the spool directory of the corresponding execution daemon. In case you have *prolog* and *epilog* scripts configured, they also need to be readable by any user who may execute jobs.

    If this violates your site's security policies you may want to set *shell_start_mode* to *script_from_stdin*. This will force Sun Grid Engine to open the job script as well as the epilogue and prologue scripts for reading into `STDIN` as root (if `sge_execd(8)` was started as root) before changing to the job owner's user account. The script is then fed into the `STDIN` stream of the command interpreter indicated by the `-S` option of the `qsub(1)` command or the *shell* parameter of the queue to be used (see above).

    Thus setting *shell_start_mode* to *script_from_stdin* also implies *posix_compliant* behavior.

Note – Feeding scripts into the `STDIN` stream of a command interpreter may cause trouble if commands like `rsh(1)` are invoked inside a job script as they also process the `STDIN` stream of the command interpreter. These problems can usually be resolved by redirecting the `STDIN` channel of those commands to come from `/dev/null` (e.g., `rsh` *host date* `< /dev/null`).

Note – Any command-line options associated with the job are passed to the executing shell. The shell will only forward them to the job if they are not recognized as valid shell options.

The default for *shell_start_mode* is *posix_compliant*.

- *prolog* – This is the executable path of a shell script that is started before execution of Sun Grid Engine jobs with the same environment setting as that for the Sun Grid Engine jobs to be started afterwards. An optional prefix `user@` specifies the user under which this procedure is to be started. This procedure is intended as a means for the Sun Grid Engine administrator to automate the execution of general site specific tasks like the preparation of temporary file systems with the need for the same context information as the job. This queue configuration entry overwrites cluster global or execution host specific *prolog* definitions (see `sge_conf(5)`).

Note – prolog is executed *exactly* as the job script. Therefore, all implications described under the parameters *shell_start_mode* and *login_shells* below apply.

  The default for *prolog* is the special value `NONE`, which prevents from execution of a prologue script. The special variables for constituting a command line are the same like in *prolog* definitions of the cluster configuration (see `sge_conf(5)`).

- *epilog* – This is the executable path of a shell script that is started after execution of Sun Grid Engine jobs with the same environment setting as that for the Sun Grid Engine jobs that has just completed. An optional prefix `user@` specifies the user under which this procedure is to be started. This procedure is intended as a means for the Sun Grid Engine administrator to automate the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job. This queue configuration entry overwrites cluster global or execution host specific *epilog* definitions (see `sge_conf(5)`).

Note – epilog is executed *exactly* as the job script. Therefore, all implications described under the parameters *shell_start_mode* and *login_shells* below apply.

  The default for *epilog* is the special value `NONE`, which prevents from execution of a epilogue script. The special variables for constituting a command line are the same like in *prolog* definitions of the cluster configuration (see `sge_conf(5)`).

- *starter_method* – The specified executable path will be used as a job starter facility responsible for starting batch jobs. The executable path will be executed instead of the configured shell to start the job. The job arguments will be passed as arguments to the job starter. The following environment variables are used to pass information to the job starter concerning the shell environment which was configured or requested to start the job.
  - *SGE_STARTER_SHELL_PATH* – This is the name of the requested shell to start the job.
  - *SGE_STARTER_SHELL_START_MODE* – This is the *configured shell_start_mode.*

    - *SGE_STARTER_USE_LOGIN_SHELL* – Set to `true` if the shell is supposed to be used as a login shell (see *login_shells* in `sge_conf(5)`).

  The *starter_method* will not be invoked for `qsh`, `qlogin`, or `qrsh` acting as `rlogin`.

- *suspend_method*, *resume_method*, *terminate_method* – These parameters can be used for overwriting the default method used by Sun Grid Engine for suspension, release of a suspension and for termination of a job. Per default, the signals `SIGSTOP`, `SIGCONT` and `SIGKILL` are delivered to the job to perform these actions. However, for some applications this is not appropriate.

  If no executable path is given, Sun Grid Engine takes the specified parameter entries as the signal to be delivered instead of the default signal. A signal must be either a positive number or a signal name with `SIG` as prefix and the signal name as printed by `kill -l` (e.g,. `SIGTERM`).

  If an executable path is given (it must be an *absolute path* starting with a "`/`") then this command together with its arguments is started by Sun Grid Engine to perform the appropriate action. The following special variables are expanded at runtime and can be used (besides any other strings which have to be interpreted by the procedures) to constitute a command line:

    - *$host* – This is the name of the host on which the procedure is started.
    - *$job_owner* – This is the user name of the job owner.
    - *$job_id* – This is the Sun Grid Engine system's unique job identification number.
    - *$job_name* – This is the name of the job.
    - *$queue* – This is the name of the queue.
    - *$job_pid* – This is the pid of the job.

- *notify* – This is the time waited between delivery of `SIGUSR1/SIGUSR2` notification signals and suspend/kill signals if job was submitted with the `qsub(1) -notify` option.
- *owner_list* – The *owner_list* names the login names (in a comma separated list) of those users who are authorized to suspend this queue (Sun Grid Engine operators and managers can suspend queues by default). It is customary to set this field for queues on interactive workstations where the computing resources are shared between interactive sessions and Sun Grid Engine jobs, allowing the workstation owner to have priority access (type *string*; default: `NONE`).
- *user_lists* – The *user_lists* parameter contains a comma-separated list of *user access lists* as described in `access_list(5)`. Each user contained in at least one of the enlisted access lists has access to the queue. If the *user_lists* parameter is set to `NONE` (the default) any user has access being not explicitly excluded via the *xuser_lists* parameter described below. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to the queue.

- *xuser_lists* – The *xuser_lists* parameter contains a comma-separated list of *user access lists* as described in `access_list(5)`. Each user contained in at least one of the enlisted access lists is not allowed to access the queue. If the *xuser_lists* parameter is set to `NONE` (the default) any user has access. If a user is contained both in an access list enlisted in *xuser_lists* and *user_lists* the user is denied access to the queue.
- *projects* – The *projects* parameter contains a comma-separated list of projects that have access to the queue. Any projects not in this list are denied access to the queue. If set to `NONE` (the default), any project has access that is not specifically excluded via the *xprojects* parameter described below. If a project is in both the *projects* and *xprojects* parameters, the project is denied access to the queue. This parameter is only available in a Sun Grid Engine, Enterprise Edition system.
- *xprojects* – The *xprojects* parameter contains a comma-separated list of projects that are denied access to the queue. If set to `NONE` (the default), no projects are denied access other than those denied access based on the *projects* parameter described above. If a project is in both the *projects* and *xprojects* parameters, the project is denied access to the queue. This parameter is only available in a Sun Grid Engine, Enterprise Edition system.
- *subordinate_list* – This is a list of Sun Grid Engine queues, residing on the same host as the configured queue, to suspend when a specified count of jobs is running in this queue. The list specification is the same as that of the *load_thresholds* parameter above; i.e., `low_pri_q=5,small_q`. The numbers denote the job slots of the queue that have to be filled to trigger the suspension of the subordinated queue. If no value is assigned a suspension is triggered if all slots of the queue are filled.

  On nodes which host more than one queue, you might want to accord better service to certain classes of jobs (e.g., queues that are dedicated to parallel processing might need priority over low priority production queues; the default is `NONE`).
- *complex_list* – This is the comma-separated list of administrator-defined complexes (see `complex(5)` for details) to be associated with the queue. Only complex attributes contained in the enlisted complexes and those from the *global*, *host*, and *queue* complex, which are implicitly attached to each queue, can be used in the *complex_values* list below.

  The default value for this parameter is `NONE`; i.e., no administrator-defined complexes are associated with the queue.
- *complex_values* – *complex_values* defines quotas for resource attributes managed via this queue. The allowed complex attributes to appear in *complex_values* are defined by *complex_list* (see above). The syntax is the same as for *load_thresholds* (see above). The quotas are related to the resource consumption of all jobs in a queue in the case of consumable resources (see `complex(5)` for details on consumable resources) or they are interpreted on a per queue slot (see *slots* above) basis in the case of non-consumable resources. Consumable resource attributes are

commonly used to manage free memory, free disk space or available floating software licenses while non-consumable attributes usually define distinctive characteristics like type of hardware installed.

For consumable resource attributes an available resource amount is determined by subtracting the current resource consumption of all running jobs in the queue from the quota in the *complex_values* list. Jobs can only be dispatched to a queue if no resource requests exceed any corresponding resource availability obtained by this scheme. The quota definition in the *complex_values* list is automatically replaced by the current load value reported for this attribute, if load is monitored for this resource and if the reported load value is more stringent than the quota. This effectively avoids oversubscription of resources.

Note – Load values replacing the quota specifications may have become more stringent because they have been scaled (see `host_conf(5)`) and/or load adjusted (see `sched_conf(5)`). The `-F` option of `qstat(1)` and the load display in the `qmon(1)` queue control dialog (activated by clicking on a queue icon while the "Shift" key is pressed) provide detailed information on the actual availability of consumable resources and on the origin of the values taken into account currently.

Note – The resource consumption of running jobs (used for the availability calculation) as well as the resource requests of the jobs waiting to be dispatched either may be derived from explicit user requests during job submission (see the `-l` option to `qsub(1)`) or from a "default" value configured for an attribute by the administrator (see `complex(5)`). The `-r` option to `qstat(1)` can be used for retrieving full detail on the actual resource requests of all jobs in the system.

For non-consumable resources Sun Grid Engine simply compares the job's attribute requests with the corresponding specification in *complex_values* taking the relation operator of the complex attribute definition into account (see `complex(5)`). If the result of the comparison is `true`, the queue is suitable for the job with respect to the particular attribute. For parallel jobs, each queue slot to be occupied by a parallel task is meant to provide the same resource attribute value.

Note – Only numeric complex attributes can be defined as consumable resources and hence non-numeric attributes are always handled on a per queue slot basis.

The default value for this parameter is `NONE`; i.e., no administrator-defined resource attribute quotas are associated with the queue.

- *calendar* – This specifies the *calendar* to be valid for this queue or contains `NONE` (the default). A calendar defines the availability of a queue depending on time of day, week, and year. Refer to `calendar_conf(5)` for details on the Sun Grid Engine calendar facility.

Note – Jobs can request queues with a certain calendar model via a `-l c=`*<cal_name>* option to `qsub(1)`.

- *initial_state* – This is defines an initial state for the queue either when adding the queue to the system for the first time or on startup of the `sge_execd(8)` on the host on which the queue resides. Possible values are:
  - *Default* – The queue is enabled when adding the queue or is reset to the previous status when `sge_execd(8)` comes up (this corresponds to the behavior in earlier Sun Grid Engine releases not supporting initial_state).
  - *Enabled* – The queue is enabled in either case. This is equivalent to a manual and explicit `qmod -e` command (see `qmod(1)`).
  - *Disabled* – The queue is disable in either case. This is equivalent to a manual and explicit `qmod -d` command (see `qmod(1)`).
- *fshare* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The functional shares of the queue (i.e. job class). Jobs executing in this queue may get functional tickets derived from the relative importance of the queue if the functional policy is activated.
- *oticket* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The override tickets of the queue (i.e. job class). Sun Grid Engine, Enterprise Edition distributes the configured amount of override tickets among all jobs executing in this queue.

Resource Limits

The first two resource limit parameters, `s_rt` and `h_rt`, are implemented by Sun Grid Engine. They define the “real time” or also called “elapsed” or “wall clock” time having passed since the start of the job. If `h_rt` is exceeded by a job running in the queue, it is aborted via the `SIGKILL` signal (see `kill(1)`). If `s_rt` is exceeded, the job is first “warned” via the `SIGUSR1` signal (which can be caught by the job) and finally aborted after the notification time defined in the queue configuration parameter *notify* (see above) has passed.

The resource limit parameters `s_cpu` and `h_cpu` are implemented by Sun Grid Engine as a job limit. They impose a limit on the amount of combined CPU time consumed by all the processes in the job. If `h_cpu` is exceeded by a job running in the queue, it is aborted via a `SIGKILL` signal (see `kill(1)`). If `s_cpu` is exceeded, the job is sent a `SIGXCPU` signal which can be caught by the job. If you want to allow a job to be “warned” so it can exit gracefully before it is killed then you should set the s_cpu limit to a lower value than h_cpu. For parallel processes, the limit is applied per slot which means that the limit is multiplied by the number of slots being used by the job before being applied.

The resource limit parameters s_vmem and h_vmem are implemented by Sun Grid Engine as a job limit. They impose a limit on the amount of combined virtual memory consumed by all the processes in the job. If h_vmem is exceeded by a job running in the queue, it is aborted via a SIGKILL signal (see kill(1)). If s_vmem is exceeded, the job is sent a SIGXCPU signal which can be caught by the job. If you wish to allow a job to be "warned" so it can exit gracefully before it is killed then you should set the s_vmem limit to a lower value than h_vmem. For parallel processes, the limit is applied per slot which means that the limit is multiplied by the number of slots being used by the job before being applied.

The remaining parameters in the queue configuration template specify per job soft and hard resource limits as implemented by the setrlimit(2) system call. See this manual page on your system for more information. By default, each limit field is set to infinity (which means RLIM_INFINITY as described in the setrlimit(2) manual page). The value type for the CPU-time limits s_cpu and h_cpu is time. The value type for the other limits is memory.

Note – Not all systems support setrlimit(2).

Note – s_vmem and h_vmem (virtual memory) are only available on systems supporting RLIMIT_VMEM (see setrlimit(2) on your operating system).

The UNICOS operating system supplied by SGI/Cray does not support the *setrlimit(2)* system call, using their own resource limit-setting system call instead. For UNICOS systems only, the following meanings apply.

- s_cpu – This is the per-process CPU time limit in seconds.
- s_core – This is the per-process maximum core file size in bytes.
- s_data – This is the per-process maximum memory limit in bytes.
- s_vmem – This is the same as s_data (if both are set the minimum is used).
- h_cpu – This is the per-job CPU time limit in seconds.
- h_data – This is the per-job maximum memory limit in bytes.
- h_vmem – This is the same as h_data (if both are set the minimum is used).
- h_fsize – This is the total number of disk blocks that this job can create.

See Also

`sge_intro(1)`, `csh(1)`, `qconf(1)`, `qmon(1)`, `qrestart(1)`, `qstat(1)`, `qsub(1)`, `sh(1)`, `nice(2)`, `setrlimit(2)`, `access_list(5)`, `calendar_conf(5)`, `sge_conf(5)`, `complex(5)`, `host_conf(5)`, `sched_conf(5)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_shepherd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sched_conf(5)

Name

`sched_conf` – Sun Grid Engine default scheduler configuration file

Description

`sched_conf` defines the configuration file format for the Sun Grid Engine system's default scheduler provided by `sge_schedd(8)`. In order to modify the configuration, use the graphical user's interface `qmon(1)` or the `-msconf` option of the `qconf(1)` command. A default configuration is provided together with the Sun Grid Engine distribution package.

Format

The following parameters are recognized by the Sun Grid Engine scheduler if present in `sched_conf`:

- *algorithm* – This allows for the selection of alternative scheduling algorithms. Currently `default` is the only allowed setting.

- *load_formula* – This is a simple algebraic expression used to derive a single weighted load value from all or part of the load parameters reported by `sge_execd(8)` for each host and from all or part of the consumable resources (see `complex(5)`) being maintained for each host. The load formula expression syntax is that of a summation weighted load values, that is:

  load_val1[*w1][{+|-}load_val2[*w2][{+|-}...]]

Note – No blanks are allowed in the load formula.

  The load values and consumable resources (load_val1, ...) are specified by the name defined in the complex (see `complex(5)`).

Note – Administrator defined load values (see the load_sensor parameter in `sge_conf(5)` for details) and consumable resources available for all hosts (see `complex(5)`) may be used as well as Sun Grid Engine default load parameters.

  The weighting factors (w1, ...) are positive integers. After the expression is evaluated for each host the results are assigned to the hosts and are used to sort the hosts corresponding to the weighted load. The sorted host list is used to sort queues subsequently.

  The default load formula is `load_avg`.

- *job_load_adjustments* – This is the load, which is imposed by the Sun Grid Engine jobs running on a system varies in time, and often, e.g. for the CPU load, requires some amount of time to be reported in the appropriate quantity by the operating system. Consequently, if a job was started very recently, the reported load may not provide a sufficient representation of the load which is already imposed on that host by the job. The reported load will adapt to the real load over time, but the period of time, in which the reported load is too low, may already lead to an oversubscription of that host. Sun Grid Engine allows the administrator to specify *job_load_adjustments* which are used in the Sun Grid Engine scheduler to compensate for this problem.

  The *job_load_adjustments* are specified as a comma separated list of arbitrary load parameters or consumable resources and (separated by an equal sign) an associated load correction value. Whenever a job is dispatched to a host by `sge_schedd(8)`, the load parameter and consumable value set of that host is increased by the values provided in the *job_load_adjustments* list. These correction values are decayed linearly over time until after *load_adjustment_decay_time* from the start the corrections reach the value 0. If the *job_load_adjustments* list is assigned the special denominator NONE, no load corrections are performed.

  The adjusted load and consumable values are used to compute the combined and weighted load of the hosts with the *load_formula* (see above) and to compare the load and consumable values against the load threshold lists defined in the queue configurations (see `queue_conf(5)`). If your *load_formula* simply consists of the

CPU load average parameter *load_avg* and if your jobs are very compute intensive, you might want to set the *job_load_adjustments* list to *load_avg=100*, which means that every new job dispatched to a host will require 100 % CPU time and thus the machine's load is instantly raised by 100.

- *load_adjustment_decay_time* – The load corrections in the "*job_load_adjustments*" list above are decayed linearly over time from the point of the job start, where the corresponding load or consumable parameter is raised by the full correction value, until after a time period of "*load_adjustment_decay_time*", where the correction becomes 0. Proper values for "*load_adjustment_decay_time*" greatly depend upon the load or consumable parameters used and the specific operating system(s). Therefore, they can only be determined on-site and experimentally. For the default *load_avg* load parameter a "*load_adjustment_decay_time*" of 7 minutes has proven to yield reasonable results.
- *maxujobs* – This is the maximum number of jobs any user may have running in a Sun Grid Engine cluster at the same time. If set to 0 (default) the users may run an arbitrary number of jobs. If the *user_sort* scheduling policy is active (see below) the scheduler allows at the most *maxujobs* in each priority group

  The *maxujobs* parameter has no effect in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine, Enterprise Edition provides more sophisticated means to control share entitlement.
- *schedule_interval* – This is the time `sge_schedd(8)` initially registers to `sge_qmaster(8)` *schedule_interval* is used to set the time interval in which `sge_qmaster(8)` sends scheduling event updates to `sge_schedd(8)`. A scheduling event is a status change that has occurred within `sge_qmaster(8)` which may trigger or affect scheduler decisions (e.g. a job has finished and thus the allocated resources are available again).

  In the Sun Grid Engine default scheduler the arrival of a scheduling event report triggers a scheduler run. The scheduler waits for event reports otherwise.

  Schedule_interval is a time value (see `queue_conf(5)` for a definition of the syntax of time values).
- *user_sort* – The Sun Grid Engine system usually schedules user jobs corresponding to a first-come-first-served policy. In case a user submits a large amount of jobs in very short time, this can lead to a rather unfair situation, because all users submitting afterwards are blocked until most of the first user's jobs are completed. Therefore, Sun Grid Engine allows to change this policy to the so called equal share sort: As soon as a user has a job running his other jobs are sorted to the end of the pending jobs list. Thus, the first jobs of all other users have comparable chances to find a queue.

Note – The equal share sort only applies within the same job priority category (refer to the `mp` option of the `qsub(1)` and `qalter(1)` commands for more information).

The default for *user_sort* is `FALSE`.

- *queue_sort_method* – This parameter determines in which order several criteria are taken into account to product a sorted queue list. Currently, two settings are valid: `seqno` and `load`. However in both cases, Sun Grid Engine attempts to maximize the number of soft requests (see `qsub(1) -s` option) being fulfilled by the queues for a particular as the primary criterion. Then, if the *queue_sort_method* parameter is set to `seqno`, Sun Grid Engine software will use the *seq_no* parameter as configured in the current queue configurations (see `queue_conf(5)`) as the next criterion to sort the queue list. The *load_formula* (see above) has only a meaning if two queues have equal sequence numbers. If *queue_sort_method* is set to `load`, the load that accords the *load_formula* is the criterion after maximizing a job's soft requests and the sequence number is only used if two hosts have the same load. The sequence number sorting is most useful if you want to define a fixed order in which queues are to be filled (e.g., the cheapest resource first).

  The default for this parameter is `load`.

- *sgeee_schedule_interval* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  This is the time period between job priority adjustments by the Sun Grid Engine, Enterprise Edition dynamic scheduler. Valid values are specified of type *time* as specified in `queue_conf(5)`.

- *halftime* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  When executing under a share based policy, Sun Grid Engine, Enterprise Edition "ages" (i.e., decreases) usage to implement a sliding window for achieving the share entitlements as defined by the share tree. The *halftime* defines the time interval in which accumulated usage will have been decayed to half its original value. Valid values are specified of type *time* as specified in `queue_conf(5)`.

- *usage_weight_list* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  Sun Grid Engine, Enterprise Edition accounts for the consumption of the resources CPU-time, memory and IO to determine the usage which is imposed on a system by a job. A single usage value is computed from these three input parameters by multiplying the individual values by weights and adding them up. The weights are defined in the *usage_weight_list*. The format of the list is:

  `cpu=`*wcpu*,`mem=`*wmem*,`io=`*wio*

  where *wcpu*, *wmem*, and *wio* are the configurable weights. The weights are real number. The sum of all tree weights should be 1.

- *compensation_factor* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  Determines how fast Sun Grid Engine, Enterprise Edition should compensate for past usage below or above the share entitlement defined in the share tree. Recommended values are between 2 and 10, where 10 means faster compensation.

- *weight_user* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The relative importance of the user shares in the functional policy. Values are of type *real*.

- *weight_project* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The relative importance of the project shares in the functional policy. Values are of type *real*.

- *weight_jobclass* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The relative importance of the job class (i.e. queue) shares in the functional policy. Values are of type *real*.

- *weight_department* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The relative importance of the department shares in the functional policy. Values are of type *real*.

- *weight_job* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The relative importance of the job shares in the functional policy. Values are of type *real*.

- *weight_tickets_functional* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The maximum number of functional tickets available for distribution by Sun Grid Engine, Enterprise Edition. Determines the relative importance of the functional policy.

- *weight_tickets_share* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The maximum number of share based tickets available for distribution by Sun Grid Engine, Enterprise Edition. Determines the relative importance of the share tree policy.

- *weight_deadline* – This parameter is only available in a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine does not support this parameter.

  The maximum number of deadline tickets available for distribution by Sun Grid Engine, Enterprise Edition. Determines the relative importance of the deadline policy.

- *schedd_job_info* – The default scheduler can keep track why jobs could not be scheduled during the last scheduler run. This parameter enables or disables the observation. The value `true` enables the monitoring; `false` turns it off.

It is also possible to activate the observation only for certain jobs. This will be done if the parameter is set to *job_list* followed by a comma-separated list of job ids.

The user can obtain the collected information with the command `qstat -j`.

Files

- `sge_schedd` configuration – *<sge_root>*/*<cell>*`/common/sched_configuration`

See Also

`sge_intro(1)`, `qalter(1)`, `qconf(1)`, `qstat(1)`, `qsub(1)`, `complex(5)`, `queue_conf(5)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_schedd(8)`, *Sun Grid Engine 5.3 Administration and User's Guide, and Sun Grid Engine Enterprise Edition 5.3 Administration and User's Guide*

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

share_tree(5)

Name

`share_tree` – Sun Grid Engine, Enterprise Edition share tree file format

Description

The share tree object is only available in case of a Sun Grid Engine, Enterprise Edition system. Sun Grid Engine has no share tree object.

The share tree defines the long-term resource entitlements of users/projects and of a hierarchy of arbitrary groups thereof.

The current share tree can be displayed via the `qconf(1) -sstree` option. The output follows the `share_tree` format description. A share tree can be created and an existing can be modified via the `-astree` and `-mstree` options to `qconf(1)`. Individual share tree nodes can be created, modified, deleted, or shown via the `-astnode`, `-dstnode`, `-mstnode`, and `-sstnode` options to `qconf(1)`.

Format

The format of a share tree file is defined as follows:

- A new node starts with the attribute `id`, and equal sign and the numeric identification number of the node. Further attributes of that node follow until another `id`-keyword is encountered.
- The attribute *childnodes* contains a comma-separated list of child nodes to this node.
- The parameter *name* refers to an arbitrary name for the node or to a corresponding user (see `user(5)`) or project (see `project(5)`) if the node is a leaf node of the share tree. The name for the root node of the tree is "Root" by convention.
- The parameter *shares* defines the share of the node among the nodes with the same parent node.
- A user leaf node named *default* can be defined as a descendant of a `project(5))` node in the share tree. The default node defines the number of shares for users who are running in the project, but who do not have a user node defined under the project. The default user node is a convenient way of specifying a single node for all users which should receive an equal share of the project resources. The *default* node may be specified by itself or with other `user(5))` nodes at the same level below a project. All users, whether explicitly specified as a user node or those which map to the *default* user node, must have a corresponding `user(5))` object defined in order to get shares. Do not configure a `user(5))` object that is explicitly named `default`.

See Also

`sge_intro(1)`, `qconf(1)`, `project(5)`, `user(5)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

user(5)

Name

`user` – Sun Grid Engine, Enterprise Edition user entry file format

Description

The user object is only available in case of a Sun Grid Engine, Enterprise Edition system. The Sun Grid Engine system has no user object.

A user entry is used in Sun Grid Engine, Enterprise Edition to store ticket and usage information on a per user basis. Maintaining user entries for all users participating in a Sun Grid Engine, Enterprise Edition system is required if Sun Grid Engine, Enterprise Edition is operated under a user share tree policy.

A list of currently configured user entries can be displayed via the `qconf(1) -suserl` option. The contents of each enlisted user entry can be shown via the `-suser` switch. The output follows the *user* format description. New user entries can be created and existing can be modified via the `-auser`, `-muser` and `-duser` options to `qconf(1)`.

Format

A user entry contains four parameters.

- *name* – This is the user name.
- *oticket* – This is the amount of override tickets currently assigned to the user.
- *fshare* – This is the current functional share of the user.
- *default_project* – This is the default project of the user.

See Also

`sge_intro(1)`, `qconf(1)`

Copyright

See sge_intro(1) for a full statement of rights and permissions.

usermapping(5)

Name

usermapping - user mapping entry file format

Description

A usermapping entry is used to define alias names for a cluster user. The user's name—as known by the sheduling system—is called *cluster user*. If the cluster name doesn't match the user account name on an execution host, the usermapping feature can solve the problem.

Each line in the usermapping entry file specifies a user name and the host(s) where he or she has an account.

You can display a list of currently configured usermapping entries via the qconf(1) -sumapl option. The contents of each listed usermapping entry can be shown via the -sumap switch. The output follows the usermapping format description. You can create or modify user entries via the -aumap, -mumap, and *-dumap* options to qconf(1).

Format

A usermapping entry contains at least three parameters.

- *cluster_user* keyword – The *cluster_user* keyword defines the cluster user name. The rest of the textline after the keyword *cluster_user* will be taken as cluster user value.
- *name* – This is the user name on an execution host.
- *hostname* – This is the name of the host where the user specified in *name* has an account. If the first character of the host name is a @ sign, the name is used to reference a hostgroup(5).

Example

The following is a typical `usermapping` entry for a cluster usermapping.

```
cluster_user office
alice wonderland, realworld, @office2
```

The entry will map the user, `office`, which is defined in the cluster system to the user, `alice`, on the hosts `wonderland` and `realworld`. The `@` sign is used to reference a host group. So the user, `office`, is mapped to `alice` on all hosts specified in the host group, `office2`. See `hostgroup(5)` to obtain more information about that.

See Also

`qconf(1)`, `hostgroup(5)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_commd(8)

Name

`sge_commd` – Sun Grid Engine communication agent

Synopsis

`sge_commd [ -S ] [ -a` *aliasfile* `] [ -dhr ] [ -help ]`
`[ -ll` *loglevel* `] [ -ml` *fname* `] [ -nd ] [ -p` *port* `]`
`[ -s` *service* `]`

Description

All network communication in a Sun Grid Engine cluster is performed via the communication daemons `sge_commd`. Client programs like `qsub(1)` or `qstat(1)` as well as Sun Grid Engine daemons such as `sge_qmaster(8)` or `sge_execd(8)` use the service provided by `sge_commd` in order to send/receive messages to/from other Sun Grid Engine components.

`sge_commd` handles an arbitrary number of concurrent synchronous or asynchronous communications. Usually one `sge_commd` is started up automatically on each host on which `sge_qmaster(8)`, `sge_execd(8)` or/and `sge_schedd(8)` are invoked. It is however possible to connect multiple hosts via one `sge_commd` or to use a `sge_commd` on a submit or administrative Sun Grid Engine host (without running one of the other Sun Grid Engine daemons) as communication agent for the Sun Grid Engine client programs invoked from that host.

Options

TABLE 29 lists and describes options associated with `sge_cmmd`.

TABLE 29 `sge_cmmd` Options

Option	Description
`-S`	Forces secure ports to be used for communication between `sge_commd` and between other Sun Grid Engine components and the `sge_commds`. This requires all Sun Grid Engine daemons to be started with root permission and the client programs to be configured set-uid root. In turn, it ensures that unauthorized communication is prohibited for non-root accounts.
`-a` *aliasfile*	A file containing Sun Grid Engine host aliases used by `sge_commd` to resolve Sun Grid Engine unique host names for all hosts in the cluster. Overrides the usage of the default *host_aliases* file under *<sge_root>*/*<cell>*`/common/host_aliases`. The host name resolving service of `sge_commd` is also used by all other Sun Grid Engine components. The file format and the implication of its usage are described in `sge_h_aliases(5)`.
`-dhr`	The host name resolving C-library functions (such as `gethostent(3)`, `gethostbyname(3)` and `gethostbyaddr(3)`) perform some kind of caching on some OS architectures. Network wide host name databases distributed by services such as DNS (Domain Name Service) and NIS are updated with a delay of several minutes. This only affects applications which repeatedly resolve host names (such as `sge_commd`). At startup of a program, the most recent information is accessed; thus commands such as `telnet(1)` or `nslookup(1)` are not affected. However, for `sge_commd` it makes no sense to resolve host names anytime (the returned information may be out of date anyway) and resolving can be an expensive operation if the network is overloaded and/or NIS or DNS servers are very busy. Therefore, `sge_commd` resolves host name information from time to time only. Yet, if host name resolving still causes problems due to network load, for example, it can be turned off with the `-dhr` switch. The administrator has to be aware, that if the host name resolving is turned off, `sge_commd` has to be restarted as soon as the host name databases change significantly.
`-help`	Prints a listing of all options.
`-ll` *loglevel*	Sets a logging level for error tracing. The error trace information is written to the file */tmp/commd/err.<pid>*. However, the directory */tmp/commd* must be present, otherwise the tracing output is discarded. At present, 255 is the only valid logging level.
`-ml` *fname*	Enables message logging to file.

TABLE 29 `sge_cmmd` Options *(Continued)*

Option	Description
`-nd`	Do not daemonize. If started with `-nd`, `sge_commd` maintains its connection to the controlling terminal and thus outputs trace information directly to the terminal from which `sge_commd` was invoked. The trace information is the same as being accessible via the `-ll` option (see above).
`-p` *port_number*	Use this TCP port for communication with other commds.
`-s` *service_name*	Use this service name and thus the associated TCP port for communication with other `commds`.

Environment Variables

TABLE 30 lists the environment variables associated with `sge_cmmd`.

TABLE 30 `sge_cmmd` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	(Does not affect the behavior of `sge_commd` but of the other Sun Grid Engine components contacting `sge_commd`.) If set, specifies the host on which the particular `sge_commd` to be used for Sun Grid Engine communication of arbitrary Sun Grid Engine client programs or daemons resides. Per default the local host is used.

Restrictions

`sge_commd` usually is invoked by a starting `sge_qmaster(8)` and `sge_execd(8)` and thus is running under root permission. If started by a normal user the `-S` switch may not be used as the secure mode requires root permission (see above).

See Also

sge_intro(1), sge_h_aliases(5), sge_execd(8), sge_qmaster(8), commdcntl(8)

Copyright

See sge_intro(1) for a full statement of rights and permissions.

sge_execd(8)

Name

sge_execd – Sun Grid Engine job execution agent

Synopsis

sge_execd [-help] [-lj *log_file*] [-nostart-*commd*]

Description

sge_execd controls the Sun Grid Engine queues local to the machine sge_execd is running on and executes/controls the jobs sent from sge_qmaster(8) to be run on these queues.

Together with sge_execd a sge_commd(8) is brought up automatically on the same machine (if not already present).

Options

TABLE 31 lists the options associated with `sge_execd`.

TABLE 31 `sge_execd` Options

Option	Description
`-help`	Prints a listing of all options.
`-lj` *log_file*	Enables job logging. All actions taken by `sge_execd` from receiving the job until returning it to `sge_qmaster(8)` are logged to the *log_file*. This feature is also available with the `sge_qmaster(8)` daemon.
`-nostart-`*commd*	Do not start up `sge_commd(8)` automatically with `sge_execd` and evaluate the `COMMD_HOST` environment variable to find the corresponding `sge_commd(8)`.

Load Sensors

If a load sensor is configured for `sge_execd` via either the global or the execution host specific cluster configuration (see `sge_conf(5)`) the executable path of the load sensor is invoked by `sge_execd` on a regular basis and delivers one or multiple load figures for the execution host (e.g. users currently logged in) or on the complete cluster (e.g. free disk space on a network wide scratch file system). The load sensor may be a script or a binary executable. In either case its handling of the `STDIN` and `STDOUT` stream and its control flow must comply to the following rules:

The load sensor has to be written as infinite loop waiting at a certain point for input from `STDIN`. If the string `quit` is read from `STDIN`, the load sensor is supposed to exit. As soon as an end-of-line is read from `STDIN` a load data retrieval cycle is supposed to start. The load sensor then performs whatever operation is necessary to compute the desired load figures. At the end of the cycle the load sensor writes the result to stdout. The format is as follows.

- A load value report starts with a line containing nothing but the word `start`.
- Individual load values are separated by newlines.
- Each load value report consists of three parts separated by colons (`:`) and containing no blanks.
- The first part of a load value information is either the name of the host for which load is reported or the special name, `global`.
- The second part is the symbolic name of the load value as defined in the host or global complex list (see `complex(5)` for details). If a load value is reported for which no entry in the host or global complex list exists, the reported load value is not used.

- The third part is the measured load value.
- A load value report ends with a line with the word end.

Environment Variables

TABLE 32 lists and describes the environment variables associated with sge_execd.

TABLE 32 sge_execd Environment Variables

Name of Variable	Description
SGE_ROOT	Specifies the location of the Sun Grid Engine standard configuration files.
SGE_CELL	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, sge_execd uses (in the order of precedence): • The name of the cell specified in the environment variable SGE_CELL, if it is set. • The name of the default cell; e.g.. default.
SGE_DEBUG_LEVEL	If set, specifies that debug information should be written to stderr. In addition the level of detail in which debug information is generated is defined.
COMMD_PORT	If set, specifies the tcp port on which sge_commd(8) is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
COMMD_HOST	If set, specifies the host on which the particular sge_commd(8) to be used for Sun Grid Engine communication of the sge_execd client resides. Only evaluated if the -nostart-*commd* option was specified at the sge_execd command line. Per default the local host is used.

Restrictions

sge_execd usually is started from root on each machine in the Sun Grid Engine pool. If started by a normal user, a spool directory must be used to which the user has read/write access. In this case only jobs being submitted by that same user are treated correctly by the system.

Files

- Sun Grid Engine global configuration – *<sge_root>*/*<cell>*`/common/configuration`
- Sun Grid Engine host-specific configuration – *<sge_root>*/*<cell>*`/common/local_conf/`*<host>*
- Default execution host spool directory – *<sge_root>*/*<cell>*`/spool/`*<host>*
- Sun Grid Engine master host file – *<sge_root>*/*<cell>*`/common/act_qmaster`

See Also

`sge_intro(1)`, `sge_conf(5)`, `complex(5)`, `sge_commd(8)`, `sge_qmaster(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_qmaster(8)

Name

`sge_qmaster` – Sun Grid Engine master control daemon

Synopsis

`sge_qmaster [ -help ] [ -lj` *log_file* `] [ -nohist ]`
`[ -nostart-`*commd* `] [ -s ]`

`sge_qmaster -show-license`

Description

`sge_qmaster` controls the overall Sun Grid Engine behavior in a cluster. For the purpose of scheduling jobs `sge_qmaster` cooperates with `sge_schedd(8)`. At startup of `sge_qmaster` `sge_commd(8)` is usually brought up automatically on the same machine (if not already present).

Options

TABLE 33 lists and describes the options associated with `sge_qmaster`.

TABLE 33 `sge_qmaster` Options

Option	Description
`–help`	Prints a listing of all options.
`–lj` *log_file*	Enables job logging. All actions taken by `sge_qmaster` from submit to job exit are logged to the *log_file*. This feature is also available with the `sge_execd(8)` daemon.
`–nohist`	During usual operation `sge_qmaster` dumps a history of queue, complex and host configuration changes to a history database. This database is primarily used with the `qacct(1)` command to allow for `qsub(1)` like `–l` *resource* requests in the `qacct(1)` command line. This switch suppresses writing to this database.
`–nostart-`*commd*	Do not start up `sge_commd(8)` automatically with `sge_qmaster`.
`–s`	Turns on `sge_qmaster` silent mode. Usually `sge_qmaster` displays a license information on startup and waits for a return to continue. With the `-s` switch, `sge_qmaster` starts up silently.

Environment Variables

TABLE 34 lists and describes the environment variables associated with `sge_qmaster`.

TABLE 34 `sge_qmaster` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `sge_qmaster` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

Restrictions

`sge_qmaster` is usually started from root on the master or shadow master machines of the cluster (refer to the *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide* for more information about the configuration of shadow master hosts). If started by a normal user, a master spool directory must be used to which the user has read/write access. In this case only jobs being submitted by that same user are treated correctly by the system.

Files

- Sun Grid Engine global configuration – *<sge_root>*/*<cell>*`/common/configuration`
- Sun Grid Engine host-specific configuration – *<sge_root>*/*<cell>*`/common/local_conf/`*<host>*
- History database – *<sge_root>*/*<cell>*`/common/history`
- `sge_qmaster` argument file – *<sge_root>*/*<cell>*`/common/qmaster_args`
- Default master spool directory – *<sge_root>*/*<cell>*`/spool`

See Also

sge_intro(1), sge_conf(5), sge_commd(8), sge_execd(8), sge_schedd(8), sge_shadowd(8), *Sun Grid Engine 5.3 Administration and User's Guide, and Sun Grid Engine Enterprise Edition 5.3 Administration and User's Guide*

Copyright

See sge_intro(1) for a full statement of rights and permissions.

sge_schedd(8)

Name

sge_schedd – Sun Grid Engine job scheduling agent

Synopsis

```
sge_schedd [ -help ]
```

Description

sge_schedd computes the scheduling decision in a Sun Grid Engine cluster. The information necessary for the decision is retrieved from sge_qmaster(8) via an event interface. After applying the scheduling algorithm, sge_schedd communicates the scheduling decision back to sge_qmaster(8) again via the Sun Grid Engine GDI. In order to trigger a sge_schedd run, sge_qmaster(8) samples changes in the cluster status and notifies sge_schedd in periodical time intervals.

Together with sge_schedd, a sge_commd(8) is brought up automatically on the same machine (if not already present).

By using the `-tsm` option of the `qconf(1)` command, `sge_schedd` can be forced to print trace messages of its next scheduling run to the file *<sge_root>*/*<cell>*`/common/schedd_runlog`. The messages indicate the reasons for jobs and queues not being selected in that run.

Option

TABLE 35 lists the option associated with `sge_schedd`.

TABLE 35 `sge_schedd` Option

Option	**Description**
`-help`	Prints the version number of the scheduler.

Environment Variables

TABLE 36 lists the environment variables associated with `sge_schedd`.

TABLE 36 `sge_schedd` Environment Variables

Name of Variable	**Description**
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell `sge_schedd` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.

Files

- `sge_schedd` spool directory – *<sge_root>*/*<cell>*`/spool/qmaster/schedd`

- `sge_schedd` trace information – *<sge_root>*/*<cell>*`/common/sched_runlog`
- `sge_schedd` configuration – *<sge_root>*/*<cell>*`/common/sched_configuration`
- See `sched_conf(5)` for details on the scheduler configuration file.

See Also

`sge_intro(1)`, `sge_gdi(3)`, `sched_conf(5)`, `sge_commd(8)`, `sge_qmaster(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_shadowd(8)

Name

`sge_shadowd` – Sun Grid Engine shadow master daemon

Synopsis

`sge_shadowd`

Description

`sge_shadowd` is a "lightweight" process which can be run on the so called shadow master hosts in a Sun Grid Engine cluster to detect failure of the current Sun Grid Engine master daemon `sge_qmaster(8)` and to start up a new `sge_qmaster(8)` on the host on which the `sge_shadowd` runs. If multiple shadow daemons are active in a cluster, they run a protocol which ensures that only one of them will start up a new master daemon.

The hosts suitable for being used as shadow master hosts must have shared root read write access to the directory *<sge_root>*/*<cell>*`/common` as well as to the master daemon spool directory (Default *<sge_root>*/*<cell>*`/spool/qmaster`). The shadow master hosts need to be contained in the file *<sge_root>*/*<cell>*`/common/shadow_masters`.

Restrictions

`sge_shadowd` may only be started from root.

Environment Variables

TABLE 37 lists and describes the environment variables associated with `sge_shadowd`.

TABLE 37 `sge_shadowd` Environment Variables

Name of Variable	Description
`SGE_ROOT`	Specifies the location of the Sun Grid Engine standard configuration files.
`SGE_CELL`	If set, specifies the default Sun Grid Engine cell. To address a Sun Grid Engine cell, `sge_shadowd` uses (in the order of precedence): • The name of the cell specified in the environment variable `SGE_CELL`, if it is set. • The name of the default cell; e.g., `default`.
`SGE_DEBUG_LEVEL`	If set, specifies that debug information should be written to `stderr`. In addition the level of detail in which debug information is generated is defined.
`COMMD_PORT`	If set, specifies the tcp port on which `sge_commd(8)` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd(8)` to be used for Sun Grid Engine communication of the `sge_shadowd` client resides. Per default the local host is used.

Files

- Default configuration directory – *<sge_root>*/*<cell>*`/common`

- Shadow master host name file – *<sge_root>*/*<cell>*`/common/shadow_masters`
- Default master daemon spool directory – *<sge_root>*/*<cell>*`/spool/qmaster`

See Also

`sge_intro(1)`, `sge_conf(5)`, `sge_commd(8)`, `sge_qmaster(8)`, *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide*, *Sun Grid Engine 5.3 Administration and User's Guide*

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.

sge_sheperd(8)

Name

`sge_shepherd` – Sun Grid Engine single job controlling agent

Synopsis

`sge_shepherd`

Description

`sge_shepherd` provides the parent process functionality for a single Sun Grid Engine job. The parent functionality is necessary on UNIX systems to retrieve resource usage information (see `getrusage(2)`) after a job has finished. In addition, the `sge_shepherd` forwards signals to the job, such as the signals for suspension, enabling, termination, and the Sun Grid Engine checkpointing signal (see `sge_ckpt(1)` for details).

The sge_shepherd receives information about the job to start from the sge_execd(8). During the execution of the job it actually starts up to five child processes. First a *prolog* script if this feature is enabled by the *prolog* parameter in the cluster configuration (see sge_conf(5)). Then, a parallel environment startup script if the job is a parallel job (see sge_pe(5) for more information). Afterwards the job itself, followed by a parallel environment shutdown procedure for parallel jobs, and finally an *epilog* script if requested by the epilog parameter in the cluster configuration. The *prolog* and *epilog* scripts, as well as the parallel environment startup and shutdown procedures, are to be provided by the Sun Grid Engine administrator and are intended for site-specific actions to be taken prior to and after execution of the actual user job.

After the job has finished and the *epilog* script is processed, sge_shepherd retrieves resource usage statistics about the job, places them in a job-specific subdirectory of the spool directory of sge_execd(8) for reporting through sge_execd(8), and finishes.

Restrictions

sge_shepherd should not be invoked manually, but only by sge_execd(8).

Files

Job specific directory – *<execd_spool>/job_dir/<job_id>*

See Also

sge_intro(1), sge_conf(5), sge_execd(8)

Copyright

See sge_intro(1) for a full statement of rights and permissions.

sgecommdcntl(8)

Name

`sgecommdcntl` – Sun Grid Engine communication agent control command

Synopsis

`sgecommdcntl [ -d | -k | -t` *level* `]`
`[ -gid` *commprocname* `] [ -h[elp] ] [ -p port ] [ -S ]`
`[ -unreg` *commprocname id* `]`

Description

`sgecommdcntl` can be used to control the behavior of `sge_commd(8)` or to retrieve information from a running `sge_commd(8)`.

Options

TABLE 38 lists and describes the options associated with `sgecommdcntl`.

TABLE 38 `sgecommdcntl` Options

Option	Description
`-d`	Dump internal structures of the running `sge_commd(8)` process to `/tmp/commd/commd.dump`. The directory `/tmp/commd` must exist and `sge_commd(8)` must have write access to it. The request is ignored otherwise. This option is mainly intended for debugging purposes. The functionality of the addressed `sge_commd(8)` is not affected.
`-k`	Kill the addressed `sge_commd(8)`. Pending communications at the time of a kill request will be discarded immediately, yet the shutdown of a `sge_commd(8)` will not leave the processes being connected to the aborting process in an inconsistent state.
`-t` *level*	`sgecommdcntl` establishes a connection to `sge_commd(8)` and displays continuous trace output corresponding the trace level specified by *level*. The output consists of a subset of the trace output displayed if `sge_commd(8)` is invoked with the `-ll` option. Currently the only trace level being supported is 255.
`-gid` *commprocname*	Retrieve communication process identification number of *commprocname*. Sun Grid Engine components which enroll to `sge_commd(8)` to be able to communicate with other Sun Grid Engine processes are registered by `sge_commd(8)` with an unique identification consisting of a name and an identification number. The identification name is identical with the name of the Sun Grid Engine component (e.g., `sge_qmaster` for `sge_qmaster(8)`). The identification number can be retrieved by the `-gid` option.
`-help`	Prints a listing of all options.

TABLE 38 `sgecommdcntl` Options *(Continued)*

Option	Description
`-p` *commdport*	Port number to be used in order to address `sge_commd(8)`.
`-S`	Forces secure ports to be used for communication between `sge_commds` and between other Sun Grid Engine components and the `sge_commds`. This requires all Sun Grid Engine daemons to be started with root permission and the client programs to be configured set-uid root. In turn, it ensures that unauthorized communication is prohibited for non-root accounts.
`-unreg` *commprocname id*	Unregister Sun Grid Engine component *commprocname* registered with Id *id* to `sge_commd(8)` (see the `-gid` above for a description of *commprocname* and *id*). To unregister a Sun Grid Engine component from `sge_commd(8)` can become necessary if a Sun Grid Engine daemon such as `sge_qmaster(8)`, `sge_execd(8)` or `sge_schedd(8)` is aborted in an uncontrolled fashion (e.g. by sending the signal `SIGKILL` via `kill(1)`) and `sge_commd(8)` denies restart of that component with the message `error enrolling to commd: COMMPROC ALREADY REGISTERED`. The registration facility of `sge_commd(8)` is used to avoid redundant Sun Grid Engine daemons running on the same host. If a Sun Grid Engine component is aborted but unable to unregister from `sge_commd(8)` the registration is kept alive until a time-out of several minutes passes or until the communication process is unregistered manually.

Environment Variables

TABLE 39 lists and describes the environment variables associated with sge`commdcntl`.

TABLE 39 `sgecommdcntl` Environment Variables

Name of Variable	Description
`COMMD_PORT`	If set, specifies the tcp port on which `sgecommdcntl` is expected to listen for communication requests. Most installations will use a services map entry instead to define that port.
`COMMD_HOST`	If set, specifies the host on which the particular `sge_commd` to be used for Sun Grid Engine communication of `sgecommdcntl` resides. Per default the local host is used.

See Also

`sge_intro(1)`, `sge_commd(8)`, `sge_execd(8)`, `sge_qmaster(8)`, `sge_schedd(8)`

Copyright

See `sge_intro(1)` for a full statement of rights and permissions.