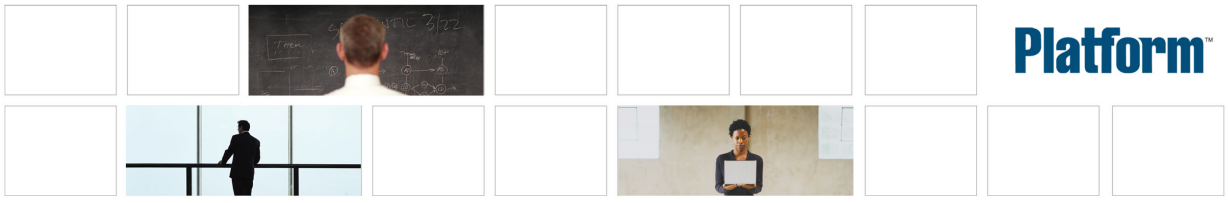
EGO Reference

Platform EGO
Version 1.2.3
January 2008



Contents

P A R T

Commands

egoconfig

Configures hosts.

Synopsis

egoconfig *subcommand* [*options*]

egoconfig -h

egoconfig -V

Description

Use the `egoconfig` command to join a host to the cluster, set the list of master candidates and the failover priority, and add hosts to or remove hosts from the ManagementHosts resource groups, or set other configuration options.

This is an administrative command. You must be logged on as cluster administrator to issue this command.

-h

Outputs command usage and exits.

-V

Prints product version to `stderr` and exits.

Subcommand synopsis

addresourceattr "[resource *resource_name*] [resourcemap *value*resource_name*] …"

join *master_name* [**-f**]

masterlist *host_name*[**,***host_name, …*]

mghost *shared_top* [**-f**]

mghost *shared_top user_account password* [**-f**]

mghost lsf [**-f**]

mghost lsf *user_account password* [**-f**]

mghost soam [**-f**]

mghost soam [*user_account*] [*password*] [**-f**]

setbaseport *base_port_no*

setlicense *license_file*

unsetmghost [**-f**]

addresourceattr "[resource *resource_name*] [resourcemap value**resource_name*] …"

Adds a resource attribute tag to the parameter EGO_LOCAL_RESOURCES in `ego.conf` on the local host. The attribute tag is later referenced when you create a resource group and want to add hosts to it that share the same resource attribute.

resource

Keyword required by EGO_LOCAL_RESOURCES to signify that the name of the resource attribute tag is boolean.

resourcemap

Keyword required by EGO_LOCAL_RESOURCES to signify that the name of the resource attribute tag is numeric. The name is preceded by a numeric value to form a name-value pair.

resource_name

Specifies the name of the resource attribute tag that identifies this type of host. For example, the resource attribute tag scvg could be later used to create a resource group for scavenging-capable hosts.

join *master_name* [-f]

For use on UNIX only. Adds the local UNIX host to the cluster that has the specified master host.

master_name

Specifies the host name of the master host.

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

masterlist *host_name*[,*host_name*, ...]

Specifies the list of master candidates, starting with the master host, and including all of the candidates, in the order of failover priority.

host_name

Specifies the name of the master host and each of the master candidates. Ensure that you do not specify compute hosts in this list.

Caution:

Be sure to include all master candidates in the list when you issue this command, as issuing this command overwrites the existing list.

mghost *shared_top* [-f]

For use on UNIX. Specifies to ignore the local configuration directory and look in the shared location for the configuration information and common files so the management hosts use a common set of files. Issuing this command adds this host to the management hosts resource group.

After issuing this command, you need to source your environment. Running this command creates an entry for the local host in `ego.cluster.`*`cluster_name`*.

shared_top

Specifies the path to the shared file location where configuration information is accessed by the management hosts in the cluster.

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

mghost *shared_top user_account password* [-f]

For use on Windows. Specifies to ignore the local configuration directory and look in the shared location for the configuration information and common files so the management hosts use a common set of files. Issuing this command adds this host to the management hosts resource group.

Issuing this command changes the behavior of Windows services on this host to run under the cluster administrator account rather than the local service account. Running this command creates an entry for the local host in `ego.cluster.`*`cluster_name`*.

shared_top

Specifies the path to the shared file location where configuration information is accessed by the management hosts in the cluster.

user_account

Specifies the user account under which to run. Must be a valid user account that has been assigned the role of CLUSTERADMIN. The format is `DOMAIN\user_name`.

password

Specifies the password to use to authenticate the user account.

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

mghost lsf [-f]

For use on UNIX. Use this command when adding Platform LSF to an existing cluster. Specifies to ignore the local LSF configuration directory and use the EGO shared directory instead. Issue this command on all management hosts so LSF can use shared location configurations.

After issuing this command, you need to source your environment.

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

mghost lsf *user_account password* [-f]

For use on Windows. Use this command when adding Platform LSF to an existing cluster. Specifies to ignore the local LSF configuration directory and use the EGO shared directory instead. Issue this command on all management hosts so LSF can use shared location configurations.

After issuing this command, you need to relaunch another DOS command prompt.

user_account

Specifies the user account under which to run. Must be a valid user account that has been assigned the role of CLUSTERADMIN. The format is `DOMAIN\user_name`.

password

Specifies the password to use to authenticate the user account.

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

setbaseport *base_port_no*

Changes the port number for the `EGO_LIM` service or daemon to the specified port number.

Caution:

Shut down the cluster before issuing this command.

The remaining system port numbers are also changed as a result of issuing this command.

base_port_no

Specifies an unused port number. The default base connection port is 7869. EGO always uses five consecutive ports starting from the base port. By default, EGO uses ports 7869-7873.

Valid base port numbers are between 1025 and 65531, inclusive.

setlicense *license_file*

For use on UNIX only. Copies the specified license file into the EGO configuration directory and updates the configuration file `ego.conf` with the name and location of the license file.

license_file

Specifies the full path to the license file including the file name.

unsetmghost [-f]

Demotes the local management host to a compute host.

Specifies to look in the local configuration directory for configuration information and common files. This command cannot be run on the master host.

Before running this command, ensure the host's lim is not running (you may need to shut down the host first). Be sure to restart the master host after running this command for the change to take effect. Running this command removes the host entry from ego.cluster.*cluster_name*.

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

egoremoverc

Prevents automatic startup of EGO on a UNIX host.

Synopsis

egoremoverc.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Prevents automatic startup of EGO on a UNIX host when a system reboot command is issued. After this script/command is issued, EGO no longer starts automatically if the host gets rebooted. In such a case, you must manually start EGO after the host has started up.

Removes the EGO links created in the system startup directory by `egosetrc`.

egosetrc

Configures automatic startup of EGO on a UNIX host.

Synopsis

egosetrc.sh

Description

Configures a UNIX host to allow automatic startup of EGO on the machine when a system reboot command is issued. Creates EGO links under the system startup directory.

This is an administrative command. You must be logged on as `root` to issue this command.

Important:

Before running this command, remove or rename EGO symbolic links under `/etc/rc.d` and `/etc/init.d`.

For ease of administration, you should enable automatic startup. This starts EGO automatically when the host restarts. After running `egosetrc.sh`, perform one of the following steps:

- Restart the machine so that EGO can start up automatically, or
- Run `egosh ego start` as root so that the cluster runs at the expected level

Note:

If you run `egosh ego start` to start the cluster, note that the EGO service under `/var/lock/subsys` is not affected. As a result, any changes to the runlevel do not automatically affect system operations (for example, dropping from level 5 to level 2 does not automatically shut down the cluster) .

If you do not configure hosts to start automatically, EGO must be started manually.

egosetsudoers

Creates an `/etc/ego.sudoers` file to determine accounts with root privileges on the UNIX host within a cluster.

Synopsis

egosetsudoers.sh

egosetsudoers.sh -f | **-p** | **-h**

Description

This command creates an `/etc/ego.sudoers` file in a UNIX host. The `/etc/ego.sudoers` file specifies accounts that are granted `root` privileges within a cluster. This file is owned by root and has the permissions set at 600.The default file is automatically configured to grant `root` privileges to the `egoadmin` account.

Note:

The cluster administrator UNIX user account is described during installation as "egoadmin" (default setting). If you indicated a different account for cluster administrator during the installation process, substitute it when you see references in the documentation to "egoadmin". Note that this is different from the cluster administrator account that you use when you log into the Platform Management Console.

This command runs `setuid` for the `egosh` command and changes the owner of `egosh` to `root`.

This is an administrative command. You must be logged on as `root` to issue this command.

You should grant `root` privileges to egoadmin so that egoadmin can start a local host in the cluster and shut down or restart any hosts in the cluster from the local host. For egoadmin or `root` to start the cluster, or start any hosts specified by name, you need to be able to run `rsh` across all hosts in the cluster without having to enter a password. See your operating system documentation for information about configuring `rsh`.

-f

Used for hosts that belong to just one cluster and when the `/etc/ego.sudoers` file already exists. Changes the cluster, preserves the existing user list, removes all other contents in the file, and saves a backup copy of the old file.

-p

Use for hosts that belong to multiple clusters and the /etc/ego.sudoers file already exists. Adds a new cluster to the path, preserves the existing user list, removes all other contents in the file, and saves a backup copy of the old file.

egosh

Launches the administrative command interface to EGO.

Synopsis

egosh

egosh *subcommand* [*options*]

egosh -h | **help**

egosh -V

Description

With no subcommands specified, launches an interactive command console to EGO. Once it is open, you can continue to issue subcommands until you close the console.

Use the `egosh` command with one or more subcommands from within a script to run commands in batch mode.

If you want to issue administrative subcommands to control EGO objects, you must first log on to EGO using the `user logon` subcommand. You are not required to log on to view EGO objects.

-h

When running the `egosh` command in shell console mode, prints command usage to `stderr` and exits.

help

When running the `egosh` command in egosh console mode, displays usage information.

-V

Prints product version to `stderr` and exits.

Subcommand synopsis

Activity synopsis

activity kill [**-s** *signal*] **-t** *activity_ID* | **-a** *alloc_ID* | **-p** *consumer_name* | **-c** *client_name*

activity list [**-l**] [**-f**] [**-t** *activity_ID* | **-a** *alloc_ID* | **-p** *consumer_name* | **-c** *client_name* |**-r** *resource*_name]

activity start -a *alloc_ID* **-r** *resource*_name [**-t** *activity_name*] [**-d** *CWD*] [**-C** *CPU_limit*] [**-F** *file_limit*] [**-D** *data_limit*] [**-S** *stack_limit*] [**-O** *core_limit*] [**-R** *rss_limit*] [**-N** *nofile_limit*] [**-A** *as_limit*] [**-e** *env_name*=*value*[;*env_name*=*value*]…] *command*

activity view [*activity_ID* …]

Allocation synopsis

alloc free -a *alloc_ID* |**-p** *consumer_name* | **-c** *client_name*

alloc list [**-l**] [**-a** *alloc_ID*] [**-p** *consumer_name*] [**-c** *client_name*] [**-r** *resource_name*] [**-t** *activity_ID*]

alloc modify -a *alloc_ID* [**-m** *min_slots*] **-M** *max_slots* [**-delta**]

alloc new -p *consumer_name* **-M** *max_slots* [**-m** *min_slots*] [**-a** *alloc_name*] [**-exclusive**] [**-g** *resource_group*] [**-s** *slots_per_host*] [**-R** *res_req*]

alloc release -a *alloc_ID* [**-block**] [**-autoadjust**] [**-modify**] *host_name***:***nslots* …

alloc unblock -a *alloc_ID* **-n** *nhosts host_name* …

alloc view [*alloc_ID* …]

Client synopsis

client list [**-l**] [**-c** *client_name*] [**-a** *alloc_ID*] [**-p** *consumer_name*] [**-r** *resource_name*] [**-t** *activity_ID*]

client reg -c *client_name* [**-d** *description*] [-t TTL]

client rm *client_name* …

client unreg

client view [*client_name* …]

client whoami

Consumer synopsis

consumer alloc [-l] [*consumer_name* …]

consumer applyresplan [-c] [-e *error_log_directory*] *file_path*

consumer list [**-l**]

consumer view [*consumer_name* …]

Debug synopsis

debug limoff *host_name* …| **all**

debug limon [**-c** *log_class*] [**-p** *timing_level*] [**-f** *log_file*] *host_name* …| all

debug pemoff *host_name* …| **all**

debug pemon [**-t**] [**-c** *log_class*] [**-p** *timing_level*] [**-f** *log_file*] [[**-o "***key***=***value***"**] …] *host_name* …| all

debug vemkdoff

debug vemkdon [**-t**] [**-c** *log_class*] [**-p** *timing_level*] [**-f** *log_file*] [[**-o "***key***=***value***"**]…]

EGO synopsis

ego elimrestart env_suffix *env_value* [-f] [*host_name* … | all]

ego execpasswd -u *user_name* **-x** *password* [**-noverify**]

ego info

ego restart [**-f**] [*host_name* …| **all**]

ego shutdown [**-f**] [*host_name* …| **all**]

ego start [**-f**] [*host_name* …| **all**]

Quit synopsis

quit

Resource synopsis

resource close [**-reclaim**] *resource_name* …

(**resource group**) | **rg** [-l] [*group_name* …]

resource list [**-l**] [**-m** | **-s** | **-t** | **-a** | **-o** *attribute*,…] [**-R** *res_req*] [*resource_name* …]

resource open *resource_name* …

resource view [*resource_name* …]

Service synopsis

service list [**-l**] [**-s** *service_name*] [**-a** *alloc_ID*] [**-p** *consumer_name*] [**-r** *resource_name*]

service start *service_name* … | all

service stop *service_name* … | all

service view [*service_name* …]

User synopsis

user add -u *user_account* **-x** *password* [**-e** *email*] [**-t** *telephone*] [**-d** *description*]

user assignrole -u *user_account* **-r** *role* [**-p** *consumer_name*]

user delete -u *user_account*

user list [**-l**]

user logoff

user logon -u *user_account* **-x** *password*

user modify -u *user_account* [**-x** *password*] [**-e** *email*] [**-t** *telephone*] [**-d** *description*]

user roles4user -u *user_account*

user users4role -r *role* [**-p** *consumer_name*]

user unassignrole -u *user_account* **-r** *role* [**-p** *consumer_name*]

user view [*user_account* …]

activity kill [-s *signal*] -t *activity_ID* | -a *alloc_ID* | -p *consumer_name* | -c *client_name*

Terminates all activities for the specified allocation, consumer or client.

-s signal

Sends a specific signal.

Signals are operating-system dependent. Refer to the list of signals available for your operating system.

-t activity_ID

Specifies the activity to terminate.

-a alloc_ID

Specifies the allocation to which this action applies.

-p consumer_name

Terminates all activities for the specified consumer.

Specify the unique consumer name, or the full path to the consumer name. Specify as many levels within the consumer tree as required to uniquely identify the consumer, as follows:

/top-level_consumer_name/. . . /leaf_consumer_name

-c client_name

Specifies the name of the client to which this action applies.

activity list [-l] [-f] [-t *activity_ID* | -a *alloc_ID* | -p *consumer_name* | -c *client_name* |-r resource_name]

Lists all activities in the cluster.

-l

Provides the same information with a longer name field, if some are truncated when -l is not specified.

-f

Lists the failed activities. These activities are in the finish state and the exit reason is not "None", "Terminated by SIGKILL", "Terminated by job controller" or "Terminated by SIGKILL, job controller does not exist or failed".

-t activity_ID

Specifies the activity to list.

-a alloc_ID

Lists all activities that belong to the specified allocation.

-p consumer_name

Lists all activities for the specified consumer.

Specify the unique consumer name, or the full path to the consumer name. Specify as many levels within the consumer tree as required to uniquely identify the consumer, as follows:

/top-level_consumer_name/. . . /leaf_consumer_name

-c client_name

Specifies the name of the client to which this action applies.

-r resource_name

Lists all the activities that are using the specified resource.

activity start -a *alloc_ID* -r *resource_name* [-t *activity_name*] [-d CWD] [-C CPU_limit] [-F *file_limit*] [-D *data_limit*] [-S *stack_limit*] [-O *core_limit*] [-R *rss_limit*] [-N *nofile_limit*] [-A *as_limit*] [-e *env_name=value*[;*env_name=value*] …] *command*

Starts an activity on the specified host.

-a alloc_ID

Specifies the allocation this activity belongs to.

-r resource_name

Specifies the host on which to start the activity.

-t activity_name

Specifies to start the activity using the name specified.

-d CWD

Specifies the current working directory from which the activity is started.

If you do not specify a directory, `/tmp` is used on UNIX systems, and `%TEMP%` is used on Windows systems.

-C CPU_limit

Specifies the maximum amount of CPU time this activity may use before being terminated by the system.

After specifying a value, specify the units for measuring CPU time:

- **s**: seconds. For example, `20s` specifies a CPU limit of 20 seconds.
- **m**: minutes. For example, **40m** specifies a CPU limit of 40 minutes.
- **h**: hours. For example, **2h** specifies a CPU limit of two hours.

-F file_limit

Specifies the maximum file size this activity may use before being terminated by the system.

After specifying a maximum file size, specify one of the following values:

- **b**: bytes. For example, `400b` specifies a limit of 400 bytes.
- **k**: kilobytes. For example, **40k** specifies a limit of 40 kilobytes.
- **m**: megabytes. For example, **4m** specifies a limit of 4 megabytes.
- **g**: gigabytes. For example, `4g` specifies a limit of 4 gigabytes.

-D data_limit

Specifies the maximum data segment size limit for each of the processes belonging to the activity. If this limit is exceeded, the activity is terminated by the system.

After specifying a value, specify the units for measuring the data limit:

- **b**: bytes. For example, `400b` specifies a limit of 400 bytes.

- **k**: kilobytes. For example, **40k** specifies a limit of 40 kilobytes.
- **m**: megabytes. For example, **4m** specifies a limit of 4 megabytes.
- **g**: gigabytes. For example, `4g` specifies a limit of 4 gigabytes.

-S stack_limit

Specifies the maximum stack segment size for each of the processes belonging to the activity. If this limit is exceeded, the activity is terminated by the system.

After specifying a value, specify the units for measuring the stack limit:

- **b**: bytes. For example, `400b` specifies a limit of 400 bytes.
- **k**: kilobytes. For example, **40k** specifies a limit of 40 kilobytes.
- **m**: megabytes. For example, **4m** specifies a limit of 4 megabytes.
- **g**: gigabytes. For example, `4g` specifies a limit of 4 gigabytes.

-O core_limit

Specifies the maximum core file size for all the processes belonging to the activity. If this limit is exceeded, the activity is terminated by the system.

After specifying a value, specify the units for measuring the core size limit:

- **b**: bytes. For example, `400b` specifies a limit of 400 bytes.
- **k**: kilobytes. For example, **40k** specifies a limit of 40 kilobytes.
- **m**: megabytes. For example, **4m** specifies a limit of 4 megabytes.
- **g**: gigabytes. For example, `4g` specifies a limit of 4 gigabytes.

-R rss_limit

Specifies the maximum resident set size, limiting physical memory usage for each process belonging to the activity. If this limit is exceeded, the activity is terminated by the system.

After specifying a value, specify the units for measuring the physical memory limit:

- **b**: bytes. For example, `400b` specifies a limit of 400 bytes.
- **k**: kilobytes. For example, **40k** specifies a limit of 40 kilobytes.
- **m**: megabytes. For example, **4m** specifies a limit of 4 megabytes.
- **g**: gigabytes. For example, `4g` specifies a limit of 4 gigabytes.

-N nofile_limit

Specifies the maximum number of open file descriptors this activity may use.

-A as_limit

Specifies the maximum process size (address space) for each process belonging to the activity.

After specifying a value, specify the units for measuring the address space limit:

- **b**: bytes. For example, `400b` specifies a limit of 400 bytes.
- **k**: kilobytes. For example, **40k** specifies a limit of 40 kilobytes.
- **m**: megabytes. For example, **4m** specifies a limit of 4 megabytes.

- **g**: gigabytes. For example, 4g specifies a limit of 4 gigabytes.

-e env_name=value …

Sets the environment variables for this activity. Specify as many environment variable/value pairs as required to define the environment.

To specify multiple environment variable/value pairs, separate the pairs with a space.

command

Specifies the command to run.

The command to run must always be specified last.

activity view [*activity_ID* …]

Displays detailed information about the activities in the cluster, including its resources, allocations, current status, start time, and so on.

activity_ID …

Specifies the ID for the activity for which you want detailed information.

alloc free -a *alloc_ID* | -p *consumer_name* | -c *client_name*

Frees all allocations for the specified consumer or client, returning resources to the cluster and removing the allocation names.

-a alloc_ID

Specifies the ID of the allocation to free.

-p consumer_name

Frees all activities for the specified consumer.

Specify the unique consumer name, or the full path to the consumer name. Specify as many levels within the consumer tree as required to uniquely identify the consumer, as follows:

/top-level_consumer_name/. . . /leaf_consumer_name

-c client_name

Specifies the name of the client to which the allocation was made.

alloc list [-l] [-a *alloc_ID*] [-p *consumer_name*] [-c *client_name*]| [-r *resource_name*] [-t *activity_ID*]

Lists all allocations in the cluster, listing the allocation ID, consumer, client, resource groups and resources used by each allocation.

-l

Provides the same information with a longer name field, if some are truncated when -l is not specified.

-a alloc _ID

Lists the allocated resources for the specified allocation.

-p consumer_name

Lists all activities for the specified consumer.

Specify the unique consumer name, or the full path to the consumer name. Specify as many levels within the consumer tree as required to uniquely identify the consumer, as follows:

/top-level_consumer_name/. . . /leaf_consumer_name

-c client_name

Lists the resources allocated to the specified client.

-r resource_name

Lists all allocations that include the specified resource.

-t activity_ID

Lists all allocations that include the specified activity.

alloc modify -a *alloc_ID* [-m *min_slots*] -M *max_slots* [-delta]

Requests an increased number of resources for an existing allocation.

-a alloc_ID

Specifies the ID of the allocation to change.

-m min_slots

Specifies the minimum number of slots to be allocated, or the minimum additional slots to be allocated, depending on if `-delta` is specified.

-M max_slots

Specifies the maximum number of slots to be allocated, or the maximum additional slots to be allocated, depending on if `-delta` is specified.

-delta

Specifies that the minimum and maximum slots requested are in addition to the existing allocation for this consumer.

alloc new -p *consumer_name* -M *max_slots* [-m *min_slots*] [-a *alloc_name*] [-exclusive] [-g *resource_group*] [-s *slots_per_host*] [-R *res_req*]

Requests a new resource allocation for the specified consumer from the specified resource group.

-p consumer_name

Specifies the consumer to allocate the resources to.

Specify the unique consumer name, or the full path to the consumer name. Specify as many levels within the consumer tree as required to uniquely identify the consumer, as follows:

/top-level_consumer_name/. . . /leaf_consumer_name

-M max_slots

Specifies the maximum number of slots to be allocated.

-m min_slots

Specifies the minimum number of slots to be allocated.

-a alloc_name

Specifies a name to identify the new allocation request.

Specify a name that is unique within the cluster. Specify up to 40 alphanumeric characters.

-exclusive

Specifies that this allocation request is for the exclusive use of these resources by this consumer.

Note that a host may still be distributed to several allocations if it appears in multiple host groups, despite indicating exclusive usage.

-g resource_group

Specifies the resource group from which to allocate resources.

-s slots_per_host

Specifies the number of slots per host required (on both single- and multi-CPU hosts).

-R res_req

Specifies the resource requirement to use to select the most appropriate host for this allocation.

Specify name value pairs for the resource requirement(s). Multiple resource requirements are separated with the characters **&&**.

Important:

If the command is issued in whole from the shell console or the requirement has white space in it , enclose the requirement in double quotation marks. For example:

```
>egosh resource list -R "select(mem>100 && it>1)"
```

If the command is issued from the egosh console, do not use quotation marks. For example:

```
>egosh
>resource list -R select(mem>100)
```

alloc release -a *alloc_ID* [-block] [-autoadjust] [-modify] *host_name*:*nslots* …

Reduces an allocation by the specified number of hosts or slots.

-a alloc_ID

Specifies the ID of the allocation from which to release the slots.

-block

Releases the slots and prevents this host from being allocated to this consumer again.

Use this option if a host is not behaving properly. You can reverse this option later using the `alloc unblock` subcommand.

-autoadjust

Automatically adjusts the allocation request to match the current number of slots. This prevents the resources from being assigned back to the current allocation.

Issuing this command without specifying a number of slots removes any unfulfilled slot requests for this allocation, and modifies the request to the current number of slots.

Use this option when you do not expect to need the slots anymore.

-modify

Automatically decrements the allocation request by the number of slots being released. The -autoadjust option takes precedence over the -modify option.

host_name:nslots …

Releases the specified number of slots from the specified hosts.

Specify the name of the host followed by the number of slots to release from that host.

To specify multiple hosts and numbers of slots, separate the host and slot combinations with a space.

alloc unblock -a *alloc_ID* -n *nhosts host_name* …

Specifies to allow blocked hosts to be allocated to this consumer again. Use this command to undo a previous `alloc release -block` subcommand.

-a alloc_ID

Specifies the ID of the allocation from which to unblock the host.

-n nhosts

Specifies the number of hosts to unblock, allowing the hosts to be allocated to this consumer again.

host_name …

Specifies the host names to unblock.

To specify multiple hosts, separate the hosts with a space.

alloc view [*alloc_ID* …]

Displays detailed information about all allocations, including the allocation ID, current users, consumer, resource groups, resource requirements, minimum and maximum slots requested, whether it has exclusive use of the host, names of the allocated hosts, and any blocked hosts.

alloc_ID …

Displays information about the specified allocation.

client list [-l] [-c *client_name*] [-a *alloc_ID*] [-p *consumer_name*] [-r *resource_name*] [-t *activity_ID*]

Displays a list of the registered clients in the cluster, and information about each client, including the host name and port number, the channel, and whether the client is connected. Client names are truncated to 12 characters.

-l

Provides the same information with a longer name field, if some are truncated when `-l` is not specified.

-c client_name

Specifies the client to which this action applies.

-a alloc_ID

Lists the client who has allocated to the specified allocation.

-p consumer_name

Lists all the clients for the specified consumer.

-r resource_name

Lists all the clients that are using the specified resource.

-t activity_ID

Lists the client that has started the specified activities.

client reg -c *client_name* [-d *description*] [-t TTL]

Registers the current EGO client with the system so that it can start sending requests to EGO for resources. Following registration, the client may be assigned allocations. The client is assigned a unique identifier such as autoAssignedClient 0 or autoAssignedClient1.

-c client_name

Specifies to register the client with a specific identifier.

Specify a name that is unique within the cluster. Specify up to 40 ASCII characters.

-d description

Specifies a description for the client being registered. This description appears with the information displayed using the `client view` subcommand. Enclose description in quotation marks if there are spaces within it.

-t TTL

Specifies the client TTL (time to live) in minutes. If the option is not set, default TTL is 900 minutes.

client rm *client_name* …

Removes and unregisters the specified client from the system. Use this command to remove a client that is not responding.

client_name …

Specifies the name of the client to be removed.

client unreg

Unregisters the current client from the system. Once this operation completes, the client can no longer request resources from EGO.

After unregistration, all allocations to this client are released.

client view [*client_name …*]

Displays a list of the registered clients in the cluster, and information about each client, including the host name and port number, the channel, and whether the client is connected.

client_name …

Specifies the name of one or more clients you want to view.

client whoami

Prints the user name associated with the current client.

consumer alloc [-l] [*consumer_name …*]

Displays allocation and demand information for each leaf consumer.

-l

Provides the same information with a longer name field, if some are truncated when -l is not specified.

consumer_name

Specifies the name of the consumer(s) for which you want to display information.

consumer applyresplan [-c] [-e *error_log_directory*] *file_path*

Applies the resource plan specified in the file path. Once you apply it, the plan is in effect immediately.

This is an administrative subcommand. You must first log on as cluster administrator or consumer administrator before you can issue this subcommand.

-c

Checks the resource plan for validity and well formedness without applying it.

-e error_log_directory

Specifies the directory where stderr with error messages outputs.

file_path

Specifies the resource plan file you want in effect immediately. The file must be XML, valid, and well-formed. If it is rejected for any reason, the previously applied resource plan stays in effect.

consumer list [-l]

Displays a list of the full paths to the consumers in the cluster, and lists the administrators assigned to each consumer.

-l

Provides the same information with a longer name field, if some are truncated when -l is not specified.

consumer view [*consumer_name …*]

Displays a list of the consumers in the cluster, and detailed information about each consumer, including the administrators assigned to that consumer and the resource policies applied to each consumer.

consumer_name

Displays information about the specified consumer.

Specify the unique consumer name without the full path or, if it is not unique, specify the full path to the consumer name.

/top-level_consumer_name/. . . /leaf_consumer_name

debug limoff *host_name … |* all

Turns off debugging of the `lim` daemon on the specified hosts.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

host_name

Turns off debugging of the `lim` daemon on the specified host.

all

Turns off debugging of the `lim` daemon on all hosts in the cluster.

debug limon [-c *log_class*] [-p *timing_level*] [-f *log_file*] *host_name …*| all

Turns on debugging of the `lim` daemon to LOG_DEBUG level on the specified host.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-c log_class

Specifies a log class, which limits the messages collected to specific types, or limits debugging to specific components. Use this option to filter out and reduce the amount of data kept.

To specify multiple log classes, separate the log classes with a space, and enclose the string in double quotes.

-p timing_level

Specifies function performance timing level to specify the number of layers of components to measure the time a process takes. Specify a number from 1 (time the process at the top component level) to 5 (time the processes at five layers depth). If no value is specified, timing is disabled.

-f log_file

Specifies the path and file name to where the log files are to be written. For example, if you specify **-f /tmp/debuglog**, the messages are logged to `/tmp/debuglog.lim.log.`*`hostname`*

If you do not specify a file name and path, defaults to the current log file.

host_name

Turns on debugging of the `lim` daemon on the specified host.

all

Turns on debugging of the `lim` daemons on all hosts in the cluster.

debug pemoff *host_name …* | all

Turns off debugging of the `pem` daemon on the specified hosts.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

host_name

Turns off debugging of the `pem` daemon on the specified host.

all

Turns off debugging of the `pem` daemon on all hosts in the cluster.

debug pemon [-t] [-c *log_class*] [-p *timing_level*] [-f *log_file*] [[-o "*key=value*"] …] *host_name* …| all

Turns on debugging of the `pem` daemon to LOG_DEBUG level on the specified host.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-t

Sets the logging level to LOG_TRACE, which logs all program steps.

-c log_class

Specifies a log class, which limits the messages collected to specific types, or limits debugging to specific components. Use this option to filter out and reduce the amount of data kept.

To specify multiple log classes, separate the log classes with a space, and enclose the string in double quotes.

-p timing_level

Specifies function performance timing level to specify the number of layers of components to measure the time a process takes. Specify a number from 1 (time the process at the top component level) to 5 (time the processes at five layers depth). If no value is specified, timing is disabled.

-f log_file

Specifies the path and file name to where the log files are to be written. For example, if you specify **-f /tmp/debuglog**, the messages are logged to `/tmp/debuglog.pem.log.`*`hostname`*

If you do not specify a file name and path, defaults to the current log file.

-o "key=value"

Specifies the debug object class and identifier. The format is *key=value*, where valid values for *key* are ACTIVITY and ALLOC and *value* is the ID of the activity or allocation.

To specify multiple key and value pairs, specify -o for each object class and separate the options with a space.

host_name

Turns on debugging of the `pem` daemon on the specified host.

all

Turns on debugging of the `pem` daemons on all hosts in the cluster.

debug vemkdoff

Turns off dynamic debugging of the EGO kernel daemon `vemkd`.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

debug vemkdon [-t] [-c *log_class*] [-p *timing_level*] [-f *log_file*] [[-o "*key=value*"] …]

Turns on dynamic debugging of the EGO kernel daemon `vemkd` to LOG_DEBUG level.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-t

Sets the logging level to LOG_TRACE, which logs all program steps.

-c log_class

Specifies a log class, which limits the messages collected to specific types, or limits debugging to specific components. Use this option to filter out and reduce the amount of data kept.

To specify multiple log classes, separate the log classes with a space and enclose the string in double quotes.

-p timing_level

Specifies function performance timing level to specify the number of layers of components to measure the time a process takes. Specify a number from 1 (time the process at the top component level) to 5 (time the processes at five layers depth). If no value is specified, timing is disabled.

-f log_file

Specifies the path to where the log files are to be written. For example, if you specify **-f /tmp/debuglog**, the messages are logged to `/tmp/debuglog.vemkd.log.`*`hostname`*

If you do not specify a file name and path, defaults to the current log file.

-o "key=value"

Specifies the debug object class and identifier. The format is *key=value*, where valid values for *key* are ACTIVITY and ALLOC and *value* is the ID of the activity or allocation.

To specify multiple key and value pairs, specify -o for each object class and separate the options with a space.

ego execpasswd -u *user_name* -x *password* [-noverify]

Registers and verifies the password for a Windows execution user account.

Registering the password allows EGO to use the account to run work on Windows hosts.

This is an administrative command. You must be cluster administrator to issue this command. In addition, to verify the password, you must be logged on to Windows as the OS account administrator, `egoadmin`.

-u user_name

Specifies the fully-qualified Windows user name of the execution account to register the password for.

-x password

Specifies the password to register for the Windows execution user account.

-noverify

Registers the password without verification. This option is required if you run this command from a UNIX host. Only a Windows host can verify this password.

ego elimrestart *env_suffix env_value* [-f] [*host_name* … | all]

Restriction:

Host scavenging is a feature for use with Platform Symphony only.

Restarts/reconfigures external load information manager(s) with an environment variable (elim.sa). Generally used for host scavenging feature. During restart, the lim passes along configuration information to the scavenging agent about the thresholds of resources that are used to evaluate trigger conditions, and whether the host is currently enabled for scavenging.

Note:

After running this command, it takes several seconds for the new configuration to take effect, dependent upon how frequently EGO refreshes host information (as set in EGO_RESOURCE_UPDATE_INTERVAL in `ego.conf`).

You must be logged on to Windows as the local systems OS account administrator, or logged on to Linux as the root OS account.

env_suffix

Always specify **SA** (scavenging agent) to indicate the host scavenging feature.

env_value

Specifies if host scavenging is currently enabled (on) or disabled (off) on this host. Specifies the thresholds of load indices used to evaluate host workload and to trigger host scavenging.

Enter the environment value in this format, delimited by commas without any spaces: *<scavenging_flag>,<user_idle_time_threshold_in_minutes>,<CPU_utilization_threshold_in_percentage>,<CPU_idle_time_threshold_in_minutes>*.

For example:

```
egosh ego elimrestart SA on,2,0.3,1.67 all
```

This example enables (turns "on") host scavenging on all hosts, sets the user idle time threshold (uit_t) to 2 minutes, CPU utilization threshold (cu_t) to 30%, and CPU idle time threshold (cit_t) to 1.67 minutes (or 100 seconds).

Note:

Threshold values are specified by numbers greater than zero. They do not need to be whole numbers.

-f

Executes command immediately without asking for confirmation. Use this option when you are issuing `egosh ego elimrestart` from within a script and do not want the script to stop running to respond to prompts.

host_name …

Specifies the name of the host or hosts on which to restart/reconfigure the external load information manager(s).

To specify multiple hosts, separate the host names with a space.

If no host name is given, then control is assumed to be local.

all

Restarts/reconfigures the external load information manager (elim) on all hosts in the cluster.

ego info

Displays information about the cluster, including the cluster name, the name of the master host, and the version of EGO.

ego restart [-f] [*host_name* … | all]

Restarts EGO on the local host, or, if issued from the master host, restarts EGO on all the hosts in the cluster. Does not affect running work or services.

This is an administrative subcommand. On UNIX, you must be logged on with `root` permissions to issue this command. On Windows, you must be logged on as cluster administrator to issue this command.

-f

Forces the action on the host without validating the configuration file. Use this option when you are issuing the command from within a script and do not want the script to stop running to respond to prompts.

host_name ...

Specifies the name of the host or hosts on which to restart EGO.

To specify multiple hosts, separate the host names with a space.

You cannot use this option from a compute host unless the master host is up and running.

all

Restarts EGO on all hosts in the cluster.

You cannot use this option from a compute host unless the master host is up and running.

ego shutdown [-f] [*host_name* … | all]

Stops EGO on the local host, or, if issued from the master host, stops EGO on all the hosts in the cluster.

This is an administrative subcommand. On UNIX, you must be logged on with `root` permissions to issue this command. On Windows, you must be logged on as cluster administrator to issue this command.

-f

Forces the action on the host without validating the configuration file. Use this option when you are issuing the command from within a script and do not want the script to stop running to respond to prompts.

host_name ...

Specifies the name of the host or hosts on which to stop EGO.

To specify multiple hosts, separate the host names with a space.

You cannot use this option from a compute host unless the master host is up and running.

all

Stops EGO on all hosts in the cluster.

Caution:

Never use this option to shut down the cluster. To shut down the entire cluster, run the egoshutdown command.

You cannot use this option from a compute host unless the master host is up and running.

ego start [-f] [*host_name* … | all]

Starts EGO on the local host or, if issued from the master host, starts EGO on all the hosts in the cluster.

This is an administrative subcommand. On UNIX, you must be logged on with `root` permissions to issue this command. On Windows, you must be logged on as cluster administrator to issue this command.

-f

Forces the action on the host without validating the configuration file. Use this option when you are issuing the command from within a script and do not want the script to stop running to respond to prompts.

host_name …

Specifies the name of the host or hosts on which to start EGO.

To specify multiple hosts, separate the host names with a space.

You cannot use this option from a compute host unless the master host is up and running.

To use this option on UNIX, you must have `root` permission on each host and have `rsh` configured for your account for each host. You may need to add an entry for the local host in the `.rhosts` file for `root`.

Note:

You cannot start EGO on a UNIX host from a Windows host, or vice versa.

all

Starts EGO on all hosts in the cluster. Use this option when you want to start the entire cluster.

You cannot use this option from a compute host unless the master host is up and running.

To use this option on UNIX, you must have `root` permission on each host and have `rsh` configured for your account. You may need to add an entry for the local host in the `.rhosts` file for `root`.

Note:

You cannot start EGO on UNIX hosts from a Windows host, or vice versa.

quit | q

Closes the interactive command console. If you are logged on to EGO, `quit` does not log you off when it closes the command console. Alias: **q**.

resource close [-reclaim] *resource_name …*

Closes a resource, preventing further allocation. Closing a resource does not change its allocation status. If the resource is currently allocated to a consumer, the resource remains allocated until the consumer returns it voluntarily. If the resource is not currently allocated to a consumer, the resource remains in its unallocated state. Existing workload finishes running before closing.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-reclaim

EGO reclaims the host before it closes; running workload terminates as per the configured grace period. The host is prevented from further allocation. If the resource is currently allocated to a consumer, it is reclaimed. Once reclaimed, it is not allocated to another consumer.

After issuing this command, the host status changes to CLOSED; the reported reason is “cluster administrator closes and reclaims host”.

resource_name …

Specifies the name of the resource or resources to close.

To close multiple resources, separate the resource names with a space.

(resource group) | rg [-l] [*group_name ...]*

Displays information about all of the resource groups in the cluster including the number of hosts in the group, the total number of slots, the number of free and allocated slots, and detailed usage information describing distribution among consumers.

rg

Is an alias to the `resource group` subcommand. You can use this as a shortcut instead of typing the full subcommand name.

- ALLOCATED: Indicates the total number of resources allocated to a consumer.
- FREE: Indicates the total number of unused resources, including unused owned and unused shared (guaranteed), as per the resource plan
- OWN: Indicates the configured ownership numbers, as per the resource plan.
- SHARE: Indicates the configured share percentage among siblings, as per the resource plan.

-l

Lists values for allocated and free slots within resource groups. Detailed usage information includes breakdown of owned, shared, and borrowed slots (both in-use and unused slots) in the cluster:

- OWN_USE: Indicates number of owned resources assigned to consumer.
- SHARE_USE: Indicates number of resources assigned to consumer from share pool.
- BORROW_USE: Indicates number of resources borrowed from other consumers.

- OWN_FREE: Indicates number of remaining (unused) owned resources as guaranteed from resource plan.
- SHARE_FREE: Indicates number of remaining (unused) share pool resources as guaranteed from resource plan.

Note:

Values for OWN_FREE and SHARE_FREE may not add up to the actual "free" or total number of resources for the resource group. Some resources reflected in the number may be reclaimed resources.

group_name

Specifies the name of the resource group for which you want information displayed. For example, `ManagementHosts`.

resource list [-l] [-m | -s | -t | -a | -o *attribute*,…] [-R *res_req*] [*resource_name …*]

Displays information about the resources in the cluster, listing each host and information about the resources on each host.

-l

Provides the same information with a longer name field, if some are truncated when -l is not specified.

-m

Displays the list of failover candidate hosts in the cluster and identifies which host is currently the master.

-s

Displays summaries of the hosts in the cluster, including information on host states and resource utilization.

-t

Displays a list of host types defined in the cluster.

-a

Displays all load indices for all resources.

-o *attribute,…*

Specifies the attributes to include in the display. Use this option to customize the output, including only those attributes you are interested in. For example:

resource list -o status,type,ncpus

Specify one (or more) of the following:

- status: Current state of the host
- type: Type of host

- ncpus: Number of CPUs as seen by EGO (value used to determine the number of slots; can be overridden by resource group configuration)
- nprocs: Number of physical processors (if ncpus defined as procs, then ncpus = nprocs)
- ncores: Number of cores per processor (if ncpus defined as cores, then ncpus = nprocs * ncores)
- nthreads: Number of threads per core (if ncpus defined as threads, then ncpus = nprocs * ncores * nthreads)
- ut: CPU utilization
- mem: Available memory
- swp: Available swap space
- pg: Paging rate
- io: Disk I/O rate
- slot: Number of slots
- freeslot: Number of free slots
- r15s: 15-second load
- r15m: 15-minute load
- r1m: 1-minute load
- model: The host model
- cpuf: The CPU factor
- maxmem: Maximum memory
- maxswp: Maximum swap space
- tmp: Available temp space
- maxtmp: Maximum space in /tmp
- ndisks: Number of local disks
- it: Idle time
- ls: Logon users
- resourceattr: Resource attributes assigned to this host
- processpri: The OS process priority of cluster workloads (either normal or lowest)

Note:

You cannot use this command option to view global ncpu settings. This information can only be viewed directly in the shared copy of `ego.conf`.

-R res_req

Displays information about the resources that match the resource requirement string specified.

Specify name value pairs for the resource requirement(s). Multiple resource requirements are separated with the characters **&&**.

Important:

If the command is issued in whole from the shell console or the requirement has white space, enclose the requirement in double quotation marks. For example:

```
>egosh resource list -R "select(mem>100 && it>1)"
```

If the command is issued from the egosh console, do not use quotation marks. For example:

```
>egosh
>resource list -R select(mem>100)
```

resource _name …

Specifies the name of the resource you want to list.

Displays information about the resource with the specified name.

resource open *resource_name* …

Opens the specified resource, allowing it to accept requests.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

resource _name …

Specifies the name of the resource or resources to open.

To open multiple resources, separate the resource names with a space.

resource view [*resource_name* …]

Displays all the information about all resources.

resource _name …

Specifies the name of the resource or resources you want to view.

Displays information about the specified resource or resources.

To view multiple resources, separate the resource names with a space.

service list [-l] [-s *service_name*] [-a *alloc_ID*] [-p *consumer_name*] [-r *resource_name*]

Lists registered service(s) defined in EGO service controller.

-l

Provides the same information with a longer name field, if some are truncated when **-l** is not specified.

-s service_name

Specifies the service to which this action applies.

-a alloc_ID

Lists all services that belong to the specified allocation.

-p consumer_name

Lists all the services for the specified consumer.

-r resource_name

Lists all the services that are using the specified resource.

service start *service_name …* | all

Starts registered service(s) defined in EGO service controller. If this is a service that is configured to start automatically, enables the service to be started automatically.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

service_name …

Starts the specified service(s).

all

Starts all registered services.

service stop *service_name …* | all

Stops registered service(s) defined in EGO service controller.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

service_name …

Stops the specified service(s).

all

Stops all registered services.

service view [*service_name ...*]

Displays registered service(s) defined in EGO service controller.

service_name …

Displays information about the specified service(s).

user add -u *user_account* -x *password* [-e *email*] [-t *telephone*] [-d *description*]

Creates a new user account in the EGO user database with the specified name.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-u user_account

Specifies the name of the user account to create.

Specify a unique name with up to 32 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key).

-x password

Specifies the password to be used to authenticate the user when this user account is accessed.

Specify one to eight alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key).

-e email

Specifies the email address of the user to whom this account belongs.

Specify up to 64 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key).

-t telephone

Specifies the telephone number of the user to whom this account belongs.

Specify up to 20 numbers and spaces.

-d description

Specifies any additional information about the user account or the user to whom this account belongs.

Specify up to 200 alphanumeric or special characters, except control characters (Ctrl + key). Enclose description in quotation marks if there are spaces within it.

user assignrole -u *user_account* -r *role* [-p *consumer_name*]

Assigns the specified role to the specified user account, and optionally specifies the consumer this role applies to.

This is an administrative subcommand. You must first log on as cluster administrator or consumer administrator before you can issue this subcommand.

-u user_account

Specifies the user account to assign the role to. The user account specified must already exist prior to issuing this command.

-r role

Specifies the role to assign. Specify one of the following:

- CLUSTER_ADMIN
- CONSUMER_ADMIN
- CONSUMER_USER

Specify CLUSTER_ADMIN to assign a user account the role of cluster administrator, with administrative authority for all consumers in the cluster. You do not need to specify a path.

Specify CONSUMER_ADMIN to assign a user account the role of consumer administrator for the specified consumer. You must specify the full path to the consumer name over which this user account should have administrative authority.

Specify CONSUMER_USER to assign a user account the role of consumer user. This role has no administrative authority, but is authorized to use resources allocated to the specified consumer. You must specify the full path to the consumer name when specifying this role.

-p consumer_name

Specifies the consumer for which this user is assigned the specified role.

Examples:

The following example assigns George Smith the role of cluster administrator:

```
egoadmin@egosh> user assignrole -u gsmith -r CLUSTER_ADMIN
```

The following example assigns Karen Dayton the role of consumer administrator for the UAT consumer and all of its descendents:

```
egoadmin@egosh> user assignrole -u kdayton -r CONSUMER_ADMIN -p testcluster/
UAT
```

The following example assigns Mark Chase the role of consumer user for the bugtest application, which is a descendent of the UAT consumer:

```
egoadmin@egosh> user assignrole -u mchase -r CONSUMER_USER -p testcluster/UAT/
bugtest
```

user delete -u *user_account*

Deletes a user account from the EGO user database.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-u user_account

Specifies the name of the user account to be deleted.

user list [-l]

Displays all user accounts in the EGO user database and the values specified for phone, email, and description.

-l

Provides the same information with a longer name field, if some are truncated when -l is not specified.

user logoff

Logs off the current user account from EGO. Logging off does not close the interactive command interface session but does prevent the user from issuing administrative subcommands.

user logon -u *user_account* -x *password*

Initiates the log on sequence to EGO, prompting for user account and password.

Note:

You are automatically logged off of EGO after 8 hours. To perform another administrative command after expiry, you are required to log on again. The logon expiry time is not configurable.

-u user_account

Specifies the EGO user account to use to log on.

-x password

Specifies the password to use to authenticate the log on sequence.

user modify -u *user_account* [-x *password*] [-e *email*] [-t *telephone*] [-d *description*]

Changes user account values to those specified for a user account defined in the EGO user database.

This is an administrative subcommand. You must first log on as cluster administrator before you can issue this subcommand.

-u user_account

Specifies the name of the user account to modify. You cannot modify the name itself.

-x password

Specifies the new password to be used to authenticate the user when this user account is accessed.

Specify one to eight alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key).

-e email

Specifies a new email address of the user to whom this account belongs.

Specify up to 64 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key).

-t telephone

Specifies the telephone number of the user to whom this account belongs.

Specify up to 20 numbers and spaces.

-d description

Specifies any additional information about the user account or the user to whom this account belongs.

Specify up to 200 alphanumeric or special characters, except control characters (Ctrl + key). Enclose description in quotation marks if there are spaces within it.

user roles4user -u *user_account*

Lists the roles assigned to a user account.

-u user_account

Specifies the user account for which to list the roles.

user users4role -r *role* [-p *consumer_name*]

Lists all user accounts in the EGO user database that have the specified role. For consumer administrators, this command also lists the consumer this user can administer. For consumer users, this command also lists the consumer to which the user has access.

-r role

-r role

Specifies the role to list all users for. Specify one of the following:

- CLUSTER_ADMIN
- CONSUMER_ADMIN

Specifies the role to list all users for. Specify one of the following:

- CLUSTER_ADMIN
- CONSUMER_ADMIN
- CONSUMER_USER

-p consumer_name

Lists all users' accounts and their roles for the specified consumer. If you specified the role CLUSTER_ADMIN, a consumer_name is not needed. If you specified either of the other two roles, a consumer_name is required.

user unassignrole -u *user_account* -r *role* [-p *consumer_name*]

Removes the specified role from the specified user account. Optionally, specifies the consumer to which this action applies or removes this role from all descendents of the specified consumer.

This is an administrative subcommand. You must first log on as cluster administrator or consumer administrator before you can issue this subcommand.

-u user_account

Specifies the user account to remove the role from.

-r role

Specifies the role to remove. Specify one of the following:

- CLUSTER_ADMIN
- CONSUMER_ADMIN
- CONSUMER_USER

-p consumer_name

Specifies the consumer for which this role is removed from the user account.

user view [*user_account* …]

Displays a list of EGO user accounts.

user_account

Specifies the name of the specific user account(s) to view.

egoshutdown

Shuts down a cluster.

Synopsis

egoshutdown.sh

egoshutdown.bat

Description

Stops all EGO components (EGO daemons or processes such as `lim`, `pem`, `vemkd`, `egosc`), stops the Platform Management Console, and, if applicable, stops various components belonging to Platform components that might be running on EGO (for example, the Platform Symphony Session Director).

Use this command when you want to shut down a cluster.

This is an administrative command. You must be a cluster administrator to issue this command. On UNIX, you must also be logged on with `root` permissions to issue this command; on Windows, you must be logged on as the OS account administrator, `egoadmin`.

You can issue this command from any host in the cluster.

egostartup

Starts all EGO components of a cluster.

Synopsis

egostartup.sh

egostartup.bat

Description

Starts all EGO components in a cluster.

This is an administrative command. On UNIX, you must also be logged on with `root` permissions to issue this command; on Windows, you must be logged on as the OS account administrator, `egoadmin`.

You cannot issue this command from a compute host unless the master host is up and running. To use this command on UNIX, you must have `root` permission on each host and have `rsh` configured for your account. You may need to add an entry for the local host in the `.rhosts` file for `root`.

You cannot start EGO on UNIX hosts from a Windows host, or vice versa.

pversions

Displays the version information for Platform products installed on a Windows host. This is a Windows command only; this command is not recognized on UNIX systems.

Synopsis

pversions [*product_name*]

pversions -h

pversions -V

Description

Displays the version and patch level of a Platform product installed on a Windows host, and the list of patches installed.

Options

product_name

Specify the Platform product or component for which you want version information. For example, specify **EGO** to see version information about EGO.

If you have a Platform product installed and running on EGO (for example, Platform Symphony or Platform LSF), you would specify the appropriate product name to see version information for each (for example, specify **Symphony** or **LSF**).

-h

Prints command usage to `stderr` and exits from the software.

-V

Prints product version to `stderr` and exits.

rfa

Transfers files between hosts.

Synopsis

rfa *subcommand* [*options*]

rfa -h

rfa -V

Description

The `rfa` subcommands are useful when transferring files between hosts during installation or package upgrades. Use this command to list files on a specified host, copy files to and from other hosts, or remove files from a host.

You must have permission to access specified directories. You must have appropriate read and/or write privileges on specified hosts.

Note:

In all cases, if no authentication information is provided (for example, user_name/password, credential), then existing credentials are automatically used. If there are no existing credentials, you are prompted for a username and password.

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

Subcommand synopsis

list -t *host_name* **-s** *remote_dir* **-p** *consumer_name* [**-c** *credential* | **-u** *user_name* **-x** *password*]

get -t *host_name* **-d** *local_destination_file* **-s** *remote_source_file* **-p** *consumer_name* [**-c** *credential* | **-u** *user_name* **-x** *password*]

put -t *host_name* **-d** *remote_destination_file* **-s** *local_source_file* **-p** *consumer_name* [**-c** *credential* | **-u** *user_name* **-x** *password*]

remove -t *host_name* **-s** *source_file* **-p** *consumer_name* [**-c** *credential* | **-u** *user_name* **-x** *password*] [**-d**]

list -t *host_name* -s *remote_dir* -p *consumer_name* [-c *credential* | -u *user_name* -x *password*]

Lists files on a specified host.

-t host_name

Name of host you want to list files for.

-s remote_dir

Absolute path to the remote directory from which you want to list files.

-p consumer_name

Name of the consumer requesting the resource allocation (which is required to run the `list` command).

-c credential

(Optional.) Authorization ID provided from current EGO logon. Credentials stored here:

- Linux
  - Directory: `/tmp`
  - Parameter: secegocc_*uid*
- Windows:
  - Directory: `%TEMP%\\secegocc_`*`osUserName`* (where `%TEMP%` is the TEMP environment variable)
  - Parameter: secegocc_*osUserName*

Note:

Use the escape character (****) before each double quoted character in the credential. For example: **Admin \"2007-01-12T02:36:44Z\"fbmTKOzDeUs1RtU +gXKJW8PYSYZZCgHXrjciZnBToGSQC/ 3tUqYXftYMzv4v1ciJ71wP+wZNeJ6cRxTmzQhOLA== \"IwN0XXwwRXaIgM5KIieZbw==**

Stored credentials expire after a certain length of time. If you do not specify an authorization ID, you need to specify a user name and password (see **-u** and **-x** options).

-u user_name

(Optional.) Fully-qualified name of the EGO user with associated permissions for the consumer requesting the resource allocation.

-x password

(Optional.) Password to register for the EGO user with associated permissions for the consumer requesting the resource allocation.

get -t *host_name* -d *local_destination_file* -s *remote_source_file* -p *consumer_name* [-c *credential* | -u *user_name* -x *password*]

Copies a file from a specified host to a local destination.

-t host_name

Name of host you want to copy a file from.

-d local_destination_file

Name of the local destination file, including its directory location, you want to copy to. Directory path must be absolute.

-s remote_source_file

Name of the remote source file, including its directory location, you want to copy. Directory path must be absolute.

-p consumer_name

Name of the consumer requesting the resource allocation (which is required to run the `get` command).

-c credential

(Optional.) Authorization ID provided from current EGO logon. Credentials stored here:

- Linux
  - Directory: `/tmp`
  - Parameter: secegocc_*uid*
- Windows:
  - Directory: `%TEMP%\\secegocc_`*`osUserName`* (where `%TEMP%` is the TEMP environment variable)
  - Parameter: secegocc_*osUserName*

Note:

Use the escape character (****) before each double quoted character in the credential. For example: **Admin \"2007-01-12T02:36:44Z\"fbmTKOzDeUs1RtU +gXKJW8PYSYZZCgHXrjciZnBToGSQC/ 3tUqYXftYMzv4v1ciJ71wP+wZNeJ6cRxTmzQhOLA== \"IwN0XXwwRXalgM5KIieZbw==**

Stored credentials expire after a certain length of time. If you do not specify an authorization ID, you need to specify a user name and password (see **-u** and **-x** options).

-u user_name

(Optional.) Fully-qualified name of the EGO user with associated permissions for the consumer requesting the resource allocation.

-x password

(Optional.) Password to register for the EGO user with associated permissions for the consumer requesting the resource allocation.

put -t *host_name* -d *remote_destination_file* -s *local_source_file* -p *consumer_name* [-c *credential* | -u *user_name* -x *password*]

Copies a file to a specified host from a local location.

-t host_name

Name of host you want to copy a file to.

-d remote_destination_file

Name of the remote destination file, including its directory location, you want to copy to. Directory path must be absolute.

-s local_source_file

Name of the local source file, including its directory location, you want to copy. Directory path must be absolute.

-p consumer_name

Name of the consumer requesting the resource allocation (which is required to run the `put` command).

-c credential

(Optional.) Authorization ID provided from current EGO logon. Credentials stored here:

- Linux
  - Directory: `/tmp`
  - Parameter: secegocc_user_id
- Windows:
  - Directory: C:`\Windows\Temp`
  - Parameter: egosec_user_id

Note:

Use the escape character (****) before each double quoted character in the credential. For example: **Admin \"2007-01-12T02:36:44Z\"fbmTKOzDeUs1RtU +gXKJW8PYSYZZCgHXrjciZnBToGSQC/ 3tUqYXftYMzv4v1ciJ71wP+wZNeJ6cRxTmzQhOLA== \"IwN0XXwwRXalgM5KIieZbw==**

Stored credentials expire after a certain length of time. If you do not specify an authorization ID, you need to specify a user name and password (see **-u** and **-x** options).

-u user_name

(Optional.) Fully-qualified name of the EGO user with associated permissions for the consumer requesting the resource allocation.

-x password

(Optional.) Password to register for the EGO user with associated permissions for the consumer requesting the resource allocation.

remove -t *host_name* -s *source_file* -p *consumer_name* [-c *credential* | -u *user_name* -x *password*] [-d]

Removes a file from a specified host.

-t host_name

Name of host you want to remove a file from.

-s source_file

Name of the source file, including its directory location, you are removing. Directory path must be absolute.

-p consumer_name

Name of the consumer requesting the resource allocation (which is required to run the `remove` command).

-c credential

(Optional.) Authorization ID provided from current EGO logon. Credentials stored here:

- Linux
  - Directory: `/tmp`
  - Parameter: secegocc_*uid*
- Windows:
  - Directory: `%TEMP%\\secegocc_`*`osUserName`* (where `%TEMP%` is the TEMP environment variable)
  - Parameter: secegocc_*osUserName*

Note:

Use the escape character (****) before each double quoted character in the credential. For example: **Admin \"2007-01-12T02:36:44Z\"fbmTKOzDeUs1RtU +gXKJW8PYSYZZCgHXrjciZnBToGSQC/ 3tUqYXftYMzv4v1ciJ71wP+wZNeJ6cRxTmzQhOLA== \"IwN0XXwwRXaIgM5KIieZbw==**

Stored credentials expire after a certain length of time. If you do not specify an authorization ID, you need to specify a user name and password (see **-u** and **-x** options).

-u user_name

(Optional.) Fully-qualified name of the EGO user with associated permissions for the consumer requesting the resource allocation.

-x password

(Optional.) Password to register for the EGO user with associated permissions for the consumer requesting the resource allocation.

-d

(Optional.) The -d flag indicates that the source file specified in the remove command is a directory. This command removes the specified directory and all its contents.

rsdeploy

Deploys and removes middleware packages.

You must be a cluster administrator to run this command.

Synopsis

rsdeploy *subcommand [options]*

rsdeploy -h

rsdeploy -V

Description

Use `rsdeploy` command to remotely deploy middleware and upgrade packages to a specified host or group of hosts, view the status of current deployment or details about past deployments, cancel a deployment, or remotely uninstall a package.

Restriction:

Middleware deployment is not intended to work on NFS installations.

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

Subcommand synopsis

add *package_name* **-p** *package_file* [**-o** *os_type*] [**-n**] [-**f**] **-u** *user_name* **-x** *password*

cancel *package_name* **-u** *user_name* **-x** *password*

install *package_name* [**-c** *consumer_name* **-r** *resource_group*] [**-r** *resource_group*] [**-t** *host_name*] [-**f**] **-u** *user_name* **-x** *password*

remove *package_name* [**-o** *os_type*] **-u** *user_name* **-x** *password*

status *package_name* [**-s all** | **allocating** | **waiting | active** | **done** | **error** | **cancelled**] **-u** *user_name* **-x** *password*

uninstall *package_name* [**-c** *consumer_name* **-r** *resource_group*] [**-r** *resource_group*] [**-t** *host_name*] **-u** *user_name* **-x** *password*

view -u *user_name* **-x** *password*

add *package_name* -p *package_file* [-o *os_type*] [-n] [-f] -u *user_name* -x *password*

Adds the middleware package to the repository server.

package_name

Specifies the package name.

The package name is used when running other `rsdeploy` commands. Enter a meaningful name, as this name, not the file name, identifies the package in the repository service.

The package name can contain up to 1024 alphanumeric characters, including the characters "." and "_". Spaces and symbols are not allowed.

Note:

A Windows limitation prohibits package names ending with ". .". For example, running the command

```
mkdir a . .
```

only creates the directory "a".

-p *package_file*

Specifies the package file to upload.

Package types can be one of the following: `tar.Z`, `tar.gz`, `tgz, gz, taz`, `tar, jar`, `tar.zip`, or `zip`.

-o *os_type*

Specifies the operating system type, as matched to the host resource type.

Typical usage is to indicate an OS type. You do not need to indicate an OS type for homogeneous clusters (where all hosts in the cluster are either Windows hosts, or all hosts are running the same Linux kernel).

-n

Indicates that package verification is not required.

By default, all packages are validated to ensure they are acceptable for deployment. Packages provided by Platform Computing do not require verification.

-f

Clears the package status for every host prior to installation.

Use this option to update the package status in cases where a manual change may have been done outside by circumventing the rsdeploy command. Ensures cached status is cleared, and the actual status is explicitly discovered. This is important as the package is installed only on those hosts that do not have the status "installed"; therefore, if a host reports an outdated status of "installed", then the package will not be installed when the command is issued.

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

cancel *package_name* -u *user_name* -x *password*

Cancels the remote deployment. Does not cancel in the middle of a package installation on a host, but stops installation on other hosts awaiting package installation.

package_name

Specifies the package name.

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

install *package_name* [-c *consumer_name -r resource_group*] [-r *resource_group*] [-t *host_name*] [-f] -u *user_name* -x *password*

Initiates deployment of the middleware package across specified hosts.

package_name

Specifies the package name.

The name was assigned when the package was first added to the repository server.

-c *consumer_name*

Specifies the consumer used to get an allocation to initiate the activity. The full consumer path is required, and must be preceded by a slash (for example, ***/ ClusterServices/EGOClusterServices***). The consumer path must be to a leaf consumer.

The consumer needs appropriate privileges/permissions to start a activity on the remote host. (Only a cluster administrator has access to all target hosts.)

Note:

If you specify a consumer name in the command, you are required to also specify a resource group.

-r *resource_group*

Specifies the resource group containing all target hosts.

-t *host_name*

Specifies host to which to install the package.

-f

Clears the package status for every host prior to installation.

Use this option to update the package status in cases where a manual change may have been done outside by circumventing the rsdeploy command. Ensures cached status is cleared, and the actual status is explicitly discovered. This is important as the package is installed only on those hosts that do not have the status "installed"; therefore, if a

host reports an outdated status of "installed", then the package will not be installed when the command is issued.

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

remove *package_name* [-o *os_type*] -u *user_name* -x *password*

Removes the middleware package from the repository server.

package_name

Specifies the package name.

-o *os_type*

Specifies the operating system type, as matched to the host resource type.

Typical usage is to indicate an OS type. You do not need to indicate an OS type for homogeneous clusters (where all hosts in the cluster are either Windows hosts, or all hosts are running the same Linux kernel).

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

status *package_name* [-s all | allocating | waiting | active | done | error | cancelled] -u *user_name* -x *password*

Gets the status of deployments, including pending and completed deployments. Lists deployment errors.

package_name

Specifies the package name.

-s

Specifies for filtering criteria for retrieving the status of deployments.

- all: Default filter. Retrieves the status on all deployments.
- allocating: Retrieves the status on deployments awaiting an allocation from EGO.
- waiting: Retrieves the status on deployments waiting for the remote agent to start.
- active: Retrieves the status on deployments with agents started on remote machines.
- done: Retrieves the status on deployments that have completed their package installations.
- error: Retrieves thee status on deployments that have received error messages.

- cancelled: Retrieves the status on deployments that were canceled.

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

uninstall *package_name* [-c *consumer_name* -r *resource_group*] [-r *resource_group*] [-t *host_name*] -u *user_name* -x *password*

Uninstalls a middleware package from the hosts.

package_name

Specifies the package name to uninstall.

-c *consumer_name*

Specifies the consumer used to get an allocation to initiate the activity. The full consumer path is required, and must be preceded by a slash (for example, ***/*ClusterServices/EGOClusterServices**). The consumer path must be to a leaf consumer.

The consumer needs appropriate privileges/permissions to start a activity on the remote host. (Only a cluster administrator has access to all target hosts.)

Note:

If you specify a consumer name in the command, you are required to also specify a resource group.

-r *resource_group*

Specifies the resource group containing all target hosts.

-t *host_name*

Specifies host to which to uninstall the package.

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

view -u *user_name* -x *password*

Lists the packages in the repository and their creation date.

-u *user_name*

Specifies the EGO user name of the component to which you are issuing this command.

-x *password*

Specifies the EGO user password of the component to which you are issuing this command.

PART II

Daemons

egosc

EGO service controller.

Synopsis

egosc [**-d** *conf_dir*]

egosc [**-C** *esc_conf_dir*]

egosc -h

egosc -V

Description

The service controller is the first service that runs on top of the EGO kernel. It functions as a bootstrap mechanism for starting the other services in the cluster. It also monitors and recovers the other services. It is somewhat analogous to `init` on UNIX systems or Service Control Manager on Windows systems. After the kernel boots, it reads a configuration file to retrieve the list of services to be started.

The service controller acts as a client to the EGO kernel, requesting resource allocations for running services and instantiating activities to host those services. It ensures that all the services are running by detecting failures and restarting service instances based on the parameter settings in the Control Policy portion of the service profile.

`egosc` is started automatically by `vemkd` and exits when `vemkd` exits. Under normal circumstances, you do not need to start it manually.

Caution:

Never start the daemon manually without checking the options: specify the `-V` option to check the version, the `-d` option to start the daemon in debug mode, or the `-C` option to validate its configuration files.

Options

-d conf_dir

Starts the daemon, reading from the EGO configuration file `ego.conf` in the specified directory, rather than from the directory set via the EGO_CONFDIR environment variable.

Use this option when starting the daemon in debug mode.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-C esc_conf_dir

Starts the daemon to validate its configuration files in the specified directory and then exits.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

Files

ego.conf

egosc reads the configuration file `ego.conf` to retrieve the location of the service controller configuration information, specified by the EGO_ESRVDIR parameter.

egosc_conf.xml

egosc reads the configuration file `egosc_conf.xml` to retrieve the following information:

ESC_PORT

The TCP port egosc uses to serve requests.

ESC_LOGDIR

The directory where egosc logs error or debug messages.

ESC_LOG_MASK

The log level used to determine the amount of detail logged.

ESC_AUDIT_LOG

Whether audit logging is turned on or off.

services files

egosc looks in the directory `EGO_ESRVDIR/esc/conf/services` to locate the services that it is to manage. Each service has a corresponding file in the `services` directory.

lim

Load information daemon or service, monitoring host load.

Synopsis

lim [**-d** *conf_dir*] [**-c** *license_file*] [**-***debug_level*]

lim -C

lim -t

lim -h

lim -V

Description

There is one lim daemon/service on every host in the cluster. Of these, one lim from the master list is elected master lim for the cluster. The master lim receives load information from the other lim daemons, and provides services to all host.

The lim does the following for the host on which it runs:

- Starts `pem` on that host
- Provides system configuration information to `vemkd`
- Monitors load and provides load information statistics to `vemkd` and users

The master lim starts vemkd and pem on the master host.

The non-master lim daemons monitor the status of the master `lim` and elect a new master (from the master list) if the current master `lim` becomes unavailable.

Collectively, the lims in the cluster coordinate the collection and transmission of load information. Load information is collected in the form of load indices.

Caution:

Never start the daemon manually without options: specify the `-V` option to check the version, the `-d` option to start the daemon in debug mode, or the `-C` option to validate its configuration files.

Options

-d conf_dir

Starts the daemon, reading from the EGO configuration file `ego.conf` in the specified directory, rather than from the directory set via the EGO_CONFDIR environment variable.

Use this option when starting the daemon in debug mode.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-c license _file

Specifies an alternative location to look for the license file rather than the default directory or the directory specified by the EGO_LICENSE_FILE environment parameter in `ego.conf`.

Specify the full path to the license file including the file name.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-debug_level

Starts the lim in debug mode. When running in debug mode, the lim uses a hard-coded port number rather than the one registered in system services.

Specify one of the following values:

-1

Starts the lim in the background, with no associated control terminal.

-2

Starts the lim in the foreground, displaying the log messages to the terminal.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-t

Displays host information, such as host type, host architecture, number of physical processors, number of cores per physical processor, number of threads per core, and license requirements.

Note:

When running Linux kernel version 2.4, you must run `lim -t` as root to ensure consistent output with other clustered application management commands (for example, output from running Platform LSF command `lshosts`).

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

Files

ego.conf

The lim reads the configuration file `ego.conf` to retrieve configuration information. `ego.conf` is a generic configuration file shared by all daemons/services and clients.

It contains configuration information and other information that dictates the behavior of the software.

Some of the parameters lim retrieves from `ego.conf` are as follows:

EGO_LIM_PORT

The TCP port the lim uses to serve all applications.

EGO_SERVERDIR

The directory used for reconfiguring the lim—where the lim binary is stored.

EGO_LOGDIR

The directory used for message logs.

EGO_LOG_MASK

The log level used to determine the amount of detail logged.

EGO_DEBUG_LIM

The log class setting for `the lim`.

EGO_LICENSE_FILE

The full path to and name of the EGO license file.

EGO_DEFINE_NCPUS

Defines whether ncpus is to be defined as procs, cores, or threads. This parameter overrides LSF_ENABLE_DUALCORE. If EGO_ENABLE_DUALCORE is set, EGO_DEFINE_NCPUS settings take precedent.

- procs (if ncpus defined as procs, then ncpus = nprocs)cores (if ncpus defined as cores, then ncpus = nprocs x ncores)
- threads (if ncpus defined as threads, then ncpus = nprocs x ncores x nthreads)

Note:

When EGO_DEFINE_NCPUS is set, run queue-length values (r1* values returned by lsload) are automatically normalized based on the set value.

If EGO_DEFINE_NCPUS is not defined, but EGO_ENABLE_DUALCORE is set, the lim reports the number of cores. If both EGO_DEFINE_NCPUS and LSF_ENABLE_DUALCORE are set, then the EGO parameter takes precedence.

EGO_ENABLE_DUALCORE

Defines if the hosts have dual cores or not. Is overridden by EGO_DEFINE_NCPUS, if set.

Note:

If EGO_DEFINE_NCPUS is not defined, but EGO_ENABLE_DUALCORE is set, the lim reports the number of cores. If both EGO_DEFINE_NCPUS and LSF_ENABLE_DUALCORE are set, then the EGO parameter takes precedence.

Customization

You can customize the `lim` by changing configuration files in EGO_CONFDIR directory. Configure `ego.cluster.<cluster_name>` to define various cluster properties such as the resources on individual hosts, the load threshold values for a host, and so on. Configure `ego.shared` to define host models read by the `lim`, or the CPU factor of individual hosts.

pem

Process execution daemon, monitoring execution.

Synopsis

pem [**-d** *conf_dir*] [**-1 | -2**]

pem -h

pem -V

Description

There is at least one `pem` daemon on every host in the cluster: on Windows, there is one per host; on Linux, there is one `pem` daemon per host, plus one for every activity running on that host—the number varies with the number of activities running at any given time.

The `pem` daemon starts all activities and monitors the activity life cycle.

The `pem` daemon is started automatically by the `lim` daemon, and exits when `lim` exits.

Caution:

Never start `pem` manually except with the `-V` option to check the version configuration.

Options

-d conf_dir

Starts the daemon, reading from the EGO configuration file `ego.conf` in the specified directory, rather than from the directory set via the EGO_CONFDIR environment variable.

Use this option when starting the daemon in debug mode.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-1

Starts the daemon in the background, with no associated control terminal. Outputs log messages into log file.

-2

Starts the daemon in debug mode. When running in debug mode, the daemon runs in the foreground, displaying the log messages. Outputs log messages to stderr.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

Files

ego.conf

`pem` reads the configuration file `ego.conf` to retrieve the location of its configuration information. `ego.conf` is a generic configuration file shared by all daemons and clients. It contains configuration information and other information that dictates the behavior of the software.

`pem` reads `ego.conf` to retrieve the following information:

EGO_PEM_PORT

The TCP port `pem` uses to serve all requests.

EGO_LOGDIR

The directory used for message logs.

EGO_TMPDIR

The temporary directory `pem` uses for persistent files. If not specified, uses your operating systerm's default temporary directory (for example, on Linux: /tmp).

EGO_LOG_MASK

The log level used to determine the amount of detail logged.

EGO_DEBUG_PEM

The log class setting for `pem`.

pim

Process information daemon, collecting resource usage of running processes.

Synopsis

pim [**-h**] [**-V**] [**-d** *conf_dir*] [*-debug_level*]

Description

The load information manager (lim) starts the pim daemon on every host participating in load sharing. The pim collects resource usage of the processes running on the local host. The information collected by the pim is used by clustered application managers (such as Platform LSF) to monitor resource consumption and enforce usage limits.

The pim updates the process information every 15 minutes unless a clustered application manager queries this information. If queried, the pim updates the process information every EGO_PIM_SLEEPTIME seconds (which is defined in the `ego.conf` file). If not defined, the default value for EGO_PIM_SLEEPTIME is 15 seconds. If the information is not queried for more than 5 minutes, the pim reverts back to the 15 minute update period.

The process information is stored in `EGO_PIM_INFODIR/pim.info.<hostname>` where EGO_PIM_INFODIR is defined in the `ego.conf` file. If this parameter is not defined, the default directory is `/tmp`. The pim daemon also reads this file when it starts up so that it can accumulate the resource usage of dead processes for existing process groups.

Note:

The pim daemon needs read access to `/dev/kmem` or its equivalent.

Options

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

-d env_dir

Read `ego.conf` from the directory `env_dir`, rather than the default directory `/etc`, or the directory specified by the $EGO_CONFDIR environment variable.

-debug_level

Starts the pim in debug mode. When running in debug mode, the pim uses a hard-coded port number rather than the one registered in system services. The debug_level option overrides the parameter EGO_LIM_DEBUG defined in `ego.conf`.

Specify one of the following values:

-1

Starts the lim in the background, with no associated control terminal.

-2

Starts the lim in the foreground, displaying the log messages to the terminal.

Files

`/etc/ego.conf` (by default) or `EGO_CONFDIR/ego.conf`

vemkd

EGO kernel daemon.

Synopsis

vemkd [**-d** *conf_dir*] [**-2**]

vemkd -C

vemkd -h

vemkd -V

Description

There is only one `vemkd` daemon in a cluster. It runs on the master host.

The `vemkd` daemon does the following:

- Starts the service controller daemon `egosc`
- Maintains security policies, allowing only authorized access
- Maintains resource allocation policies, distributing resources accordingly
- Serves as an information center where clients can query information about the cluster

The `vemkd` daemon is started automatically by the `lim` daemon, and exits when `lim` exits.

Caution:

Never start `vemkd` manually except with the `-V` option to check the version configuration.

Options

-d conf_dir

Starts the daemon, reading from the EGO configuration file `ego.conf` in the specified directory, rather than from the directory set via the EGO_CONFDIR environment variable.

Use this option when starting the daemon in debug mode.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-2

Starts the daemon in debug mode. When running in debug mode, the daemon runs in the foreground, displaying the log messages.

Caution:

Never start the daemon manually unless directed to do so by Platform Technical Support.

-C

Checks the configuration of the daemon and then exits.

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

EGO client

You can run an EGO client from any host to interact with `vemkd`, provided the client host has the environment set correctly with the EGO_CONFDIR environment variable pointing to the correct `ego.conf` file.

When the client has established a connection with `vemkd`, it can query general information about the cluster. To perform any control operations, the client needs to be logged on as an EGO user. You need to log on with at least consumer user access to request a resource from EGO, and with cluster administrator privileges to perform cluster-wide maintenance.

Files

ego.conf

`vemkd` reads the configuration file `ego.conf` to retrieve the location of its configuration information. `ego.conf` is a generic configuration file shared by all daemons and clients. It contains configuration information and other information that dictates the behavior of the software.

`vemkd` reads `ego.conf` to retrieve the following information:

EGO_KD_PORT

The TCP port `vemkd` uses to serve all requests.

EGO_WORKDIR

The directory used for data persistence.

EGO_LOGDIR

The directory used for message logs.

EGO_LOG_MASK

The log level used to determine the amount of detail logged.

EGO_DEBUG_KD

The log class setting for `vemkd`.

users.xml

Defines the EGO user accounts for the cluster.

P A R T

III

Resources

Host properties

Property	Description
Host Name	The name of the host.
Status	The current state of the host: OK, Unavailable, or Closed.
Type (Host Type)	The type of host you have. For example, LINUX86.
CPUs (Number of CPUs)	The number of CPUs you have specified for your host.
CPU Util	The current CPU utilization of your host in %.
Mem (Available Memory)	An estimate of the real memory currently available to user processes. This represents the approximate size of the largest process that could be started on a host without causing the host to start paging.
Swap (Available Swap)	The currently available virtual memory (swap space) in MB. This represents the largest process that can be started on the host.
Pg (Paging Rate)	The virtual memory paging rate in pages per second. This index is closely tied to the amount of available RAM memory and the total size of the processes running on a host; if there is not enough RAM to satisfy all processes, the paging rate is high.
I/O (Disk I/O Rate)	The I/O throughput to disks attached directly to this host, in KB per second. This rate does not include I/O to disks that are mounted from other hosts.
Slots (Number of Slots)	The number of slots you have specified for this host.
Free Slots (Number of Free Slots)	The number of slots available to run workload units at this time.
15s Load (15-Second Load)	The load this host carries, averaged over the last 15 seconds. The load is the average number of processes using the CPU during a given time interval.
15m Load (15-Minute Load)	The load this host carries, averaged over the last 15 minutes. The load is the average number of processes using the CPU during a given time interval.
1m Load (1-Minute Load)	The load this host carries, averaged over the last minute. The load is the average number of processes using the CPU during a given time interval.
Model (Host Model)	The model of your host. For example, Intel_EM64T.
CPU Factor	The speed of the host's CPU relative to other hosts in the cluster. If one processor is twice the speed of another, its CPU factor should be twice as large. The CPU factors are defined by the administrator. For multiprocessor hosts, the CPU factor is the speed of a single processor; the system automatically scales the host CPU load to account for additional processors.
Max Mem	The maximum RAM available.

Property	Description
Max Swap	The maximum swap space on your host.
Temp (Available Temp)	The space available in MB on the file system that contains the temporary directory.
Max Temp	Maximum space in `/tmp`.
Disks	Number of local disks on your host.
It (Idle Time)	The number of time in minutes that a host has been idle. On a UNIX host, it is the amount of time since the keyboard has been touched on all logged in sessions. On a Windows host, it is the amount of time a screen saver has been active.
Users (Login Users)	The number of current users logged in to the system.
Resources	If you see mg, the host is a management host.

Load indices

Index	Measures	Units	Direction	Averaged over	Update Interval
status	host status	string			15 seconds
r15s	run queue length	processes	increasing	15 seconds	15 seconds
r1m	run queue length	processes	increasing	1 minute	15 seconds
r15m	run queue length	processes	increasing	15 minutes	15 seconds
ut	CPU utilization	percent	increasing	1 minute	15 seconds
pg	paging activity	pages in + pages out per second	increasing	1 minute	15 seconds
ls	logins	users	increasing	N/A	30 seconds
it	idle time	minutes	decreasing	N/A	30 seconds
swp	available swap space	MB	decreasing	N/A	15 seconds
mem	available memory	MB	decreasing	N/A	15 seconds
tmp	available space in temporary file system	MB	decreasing	N/A	120 seconds
io	disk I/O	KB per second	increasing	1 minute	15 seconds

CPU run queue lengths (r15s, r1m, r15m)

The r15s, r1m and r15m load indices are the 15-second, 1-minute and 15-minute average CPU run queue lengths. This is the average number of processes ready to use the CPU during the given interval.

On UNIX, run queue length indices are not necessarily the same as the load averages printed by the uptime(1) command; uptime load averages on some platforms also include processes that are in short-term wait states (such as paging or disk I/O).

Effective run queue length

On multiprocessor systems, more than one process can execute at a time. The run queue value on multiprocessor systems is scaled to make the CPU load of uniprocessors and multiprocessors comparable. The scaled value is called the effective run queue length.

Normalized run queue length

The CPU run queue length is adjusted based on the relative speeds of the processors (the CPU factor). The normalized run queue length is adjusted for both number of processors and CPU speed. The host with the lowest normalized run queue length runs a CPU-intensive job the fastest.

CPU utilization (ut)

The ut index measures CPU utilization, which is the percentage of time spent running system and user code. A host with no process running has a ut value of 0 percent; a host on which the CPU is completely loaded has a ut of 100 percent.

Paging rate (pg)

The pg index gives the virtual memory paging rate in pages per second. This index is closely tied to the amount of available RAM memory and the total size of the processes running on a host; if there is not enough RAM to satisfy all processes, the paging rate is high. Paging rate is a good measure of how a machine responds to interactive use; a machine that is paging heavily feels very slow.

Login sessions (ls)

The ls index gives the number of users logged in. Each user is counted once, no matter how many times they have logged into the host.

Interactive idle time (it)

On UNIX, the it index is the interactive idle time of the host, in minutes. Idle time is measured from the last input or output on a directly attached terminal or a network pseudo-terminal supporting a login session. This does not include activity directly through the X server such as CAD applications or emacs windows, except on Solaris and HP-UX systems.

On Windows, the it index is based on the time a screen saver has been active on a particular host.

Temporary directories (tmp)

The tmp index is the space available in MB on the file system that contains the temporary directory:

- `/tmp` on UNIX
- `C:\temp` on Windows

Swap space (swp)

The swp index gives the currently available virtual memory (swap space) in MB. This represents the largest process that can be started on the host.

Memory (mem)

The mem index is an estimate of the real memory currently available to user processes. This represents the approximate size of the largest process that could be started on a host without causing the host to start paging.

Free memory is calculated as a sum of physical free memory, cached memory, buffered memory and an adjustment value.

I/O rate (io)

The io index measures I/O throughput to disks attached directly to this host, in KB per second. It does not include I/O to disks that are mounted from other hosts.

-R res_req

Selects the most appropriate resource for a particular purpose.

Synopsis

-R "select(*select_string***)"**

-R "select(*select_string***) order(***order_string***)"**

Description

A resource requirement string describes the criteria for defining a set of resources.

The entire resource requirement string cannot contain more than 512 characters.

If the characters "-" or "." form a part of the host or resource name, enclose the name using single quotation marks (for example, when a full host name is used, such as `'gr4e01.domain.name.com'`).

Examples of proper quotation mark usage include the following:

```
egosh resource list -R "select('host1.domain.name.com')"
egosh resource list -R "select('host1-1')"
egosh resource list -R "select('host1-1' ||'host1-2')"
```

Important:

If the command is issued in whole from the shell console or the requirement has white space, enclose the requirement in double quotation marks. For example:

```
egosh resource list -R "select(mem>100)"
```

If the command is issued from the egosh console, quotation marks are optional. For example:

```
egosh> resource list -R select(mem>100)
```

Options

select(select_string)

Specifies the criteria for selecting the resources. The selection string filters out the resources that do not meet the criteria, resulting in a list of one or more eligible resources.

The parentheses must be typed as shown.

order(order_string)

Specifies the sort order for selecting the best resource from the list of eligible resources. The most appropriate resource is listed first, the least is listed last.

The parentheses must be typed as shown.

select

The selection string excludes unsuitable resources by specifying the characteristics a resource must have to match a resource requirement.

select synopsis

select(*expression***)select(***expression operator expression***)select((***expression operator expression***)** *operator expression***)**

select description

The selection string is a logical expression used to select one or more resources to match one or more criteria. Any resource that satisfies the criteria is selected.

The selection string is used in many ways to select resources.

- To define resource groups in EGO
- By Platform Symphony application profiles to define the resources to run SOA workload
- To define resource requirements for both jobs and queues in Platform LSF

The resource selection string uses values for host_name, model, type, and/or resources as selection string expressions. These can be seen in the output of `egosh resource view`. When entering a resource requirement string in EGO, omit the operator and use only the value. For example:

Incorrect syntax: `select(type==linux86)`

Correct syntax: `select(linux86)`

Other examples of correct syntax:

```
select(linux86 && maxmem > 500)
select(maxmem > 2046 && LINUX86) => ib06b09
select(maxmem > 2046 && mg) => ib06b09
select(NTX86) =>host1
```

When you input a string, ensure that you use valid characters. This requirements applies to all resource requirement strings. Valid characters include the following: **a-z A-Z 0-9 * / ! ( ) . | \ ^ & $ # @ ~ % `**

select expression synopsis

resource_name operator value

resource_name

Specifies the name of the resource to use as selection criteria.

You can specify a static resource or a load index, depending on the purpose of the selection string.

operator

The following operators can be used in selection strings. If you are using the selection string in an XML format, you must use the applicable escape characters in the XML Equivalence column. The operators are listed in order of decreasing precedence:

Operator	XML Equivalent	Syntax	Meaning
!	n/a	!a	Logical NOT: 1 if a==0, 0 otherwise
*	n/a	a*b	Multiply a and b

Operator	XML Equivalent	Syntax	Meaning
/	n/a	a / b	Divide a by b
+	n/a	a+b	Add a and b
-	n/a	a-b	Subtract b from a
>	&gt;	a > b	1 if a is greater than b, 0 otherwise
<	&lt;	a < b	1 if a is less than b, 0 otherwise
>=	>=	a >= b	1 if a is greater than or equal to b, 0 otherwise
<=	<=	a <= b	1 if a is less than or equal to b, 0 otherwise
==	n/a	a == b	1 if a is equal to b, 0 otherwise
!=	n/a	a != b	1 if a is not equal to b, 0 otherwise
&&	&&	a && b	Logical AND: 1 if both a and b are non-zero, 0 otherwise
||	n/a	a || b	Logical OR: 1 if either a or b is non-zero, 0 otherwise

value

Specifies the value to be used as criteria for selecting a resource. Value can be numerical, such as when referring to available memory or swap space, or it can be textual, such as when referring to a specific type of host.

Simple selection expression using static resources

The following example selects resources with total memory greater than 500MB:

```
select(maxmem > 500)
```

Compound selection expression using static resources

The following example selects resources with total memory greater than 500MB and total swap space greater than or equal to 300MB:

```
select(maxmem > 500 && maxswp >=300)
```

Compound selection expression with precedence for dynamic load indices

The following example selects resources with available memory greater than 500MB and available swap space greater than or equal to 300MB, or at least 1000MB temporary disk space:

```
select((mem > 500 && swp >=300) || tmp >= 1000)
```

order

Sorts the selected resources into an order of preference according to the values of the resources.

order synopsis

order(*expression***)**

order description

The order string acts on the results of a selection string, sorting the selected resources to identify the most favorable resources, and eliminate the least desirable resources.

Resources are sorted into ascending order based on a load index or the result of an arithmetical expression.

The result orders the resources from smallest to largest.

order expression synopsis

order(*load_index***)**

order(*arithmetic_expression***)**

load_index

Specify the load index to use as the criteria for sorting resources.

Specify any built-in or external load index.

arithmetic _expression

Specify an arithmetic expression that expresses the desired outcome.

You can create an expression using load indices, operators, and numbers.

Use the following operators:

+

add

–

subtract

*****

multiply

/

divide

Order resources based on CPU utilization

The following example orders the selected resources based on their CPU utilization, ordering least utilized hosts first:

```
order(ut)
```

Order resources based on available swap space

The following example orders the selected resources based on their available swap space, but sorts by descending order, ordering hosts with the greatest amount of available swap space first:

```
order(0-swp)
```

-R res_req

PART

IV

Events, Environment Variables, and Audit Logs

EGO events

EGO events can be monitored and used to trigger actions automatically.

EGO events belong to the following categories:

- System events, which identify occurrences within the cluster
- Platform Management Console events, which identify occurrences that affect the web server or the Platform Management Console itself.

System events

Event name	Default level	Triggered when …
SYS_CLS_UNLICENSED • Component name: ego_lim • Returned integer: 0	Error	The cluster is not licensed
SYS_HOST_CLOSED • Component name: ego_vemkd • Returned integer: 6	Warning	A host is closed
SYS_HOST_UNAVAIL • Component name: ego_vemkd • Returned integer: 4	Warning	A host becomes unavailable
SYS_PEM_DOWN • Component name: ego_lim • Returned integer: 2	Error	Local pem goes down
SYS_PEM_UP • Component name: ego_pem • Returned integer: 12	Info	Local pem is started
SYS_SVC_DOWN • Component name: ego_sc • Returned integer: 7	Error	A system service goes down
SYS_SVC_UP • Component name: ego_sc • Returned integer: 8	Info	A system service is started

Event name	Default level	Triggered when …
SYS_SVC_INST_DOWN • Component name: ego_sc • Returned integer: 9	Error	An instance of a system service goes down
SYS_SVC_INST_UP • Component name: ego_sc • Returned integer: 10	Info	An instance of a service is started
SYS_VEMKD_DOWN • Component name: ego_lim • Returned integer: 1	Error	vemkd goes down
SYS_VEMKD_UP • Component name: ego_vemkd • Returned integer: 11	Info	vemkd is started

Platform Management Console events

Event name	Default level	Triggered when …
SYS_GUI_CPU_HI_WATER_MARK • Component name: GUI • Returned integer: 3	Warning	The web server host utilization exceeds the threshold set for `CPU_HIGH_MARK` in `wsm.conf`
SYS_GUI_MEMORY_HI_WATER_MARK • Component name: GUI • Returned integer: 2	Warning	The web server memory usage exceeds the threshold set for `MEM_HIGH_MARK` in `wsm.conf`

EGO environment variables

Environment variables are used to set the environment for commands, daemons and processes.

On UNIX, the EGO environment variables are set by the script `profile.ego` or `cshrc.ego`.

On Windows, the EGO environment variables are set by the installer.

Environment variables are primarily used internally by the software, but can be used as shortcuts to locate a particular directory.

Environment variable	Description	Default Value
EGO_BINDIR	The directory where commands are installed. Added to PATH on Linux and Path on Windows.	On Linux: *EGO_TOP*/1.2/platform/bin On Windows: *EGO_TOP*\1.2\bin
EGO_CONFDIR	The directory containing the path to the EGO configuration file `kernel/conf/ego.conf`.	On Linux: *EGO_CONFDIR*/kernel/conf or *shared_dir*/kernel/conf On Windows: *EGO_CONFDIR*\kernel\conf or *shared_dir*\kernel\conf
EGO_ESRVDIR	The directory where EGO service configuration files are stored.	On Linux: *EGO_TOP*/eservice or /*shared_dir*/eservice On Windows: *EGO_TOP*\eservice or *shared_dir*\eservice
EGO_LIBDIR	The directory where the EGO libraries are installed, added to LD_LIBRARY_PATH on Linux, added to Path on Windows.	On Linux: *EGO_TOP*/1.2/*platform*/lib On Windows: *EGO_TOP*\1.2\lib
EGO_LOCAL_CONFDIR	This is the local configuration directory.	On Linux: *EGO_TOP*/kernel/conf On Windows: *EGO_TOP*\kernel\conf

Environment variable	Description	Default Value
EGO_SERVERDIR	The directory where the EGO server binaries and shell scripts are installed, added to PATH on Linux and Path on Windows.	On Linux: *EGO_TOP*/1.2/*platform*/etc On Windows: *EGO_TOP*\1.2\etc

Note:

$*EGO_TOP* is the directory where EGO is installed, and *platform* represents the operating system. For example, for Linux: `linux2.4-glibc2.3-x86`

The most important environment variable to be set is EGO_CONFDIR, which, if not set in the current logon session, may prevent a user from running EGO clients.

EGO audit logs

EGO events related to consumers and services, users, and core operations can be collected and stored in audit logs.

Specify the location of audit log files with the parameter EGO_AUDIT_LOGDIR in `ego.conf`.

Component/ object	Logged event/action	Audit log file name

EGO service	• Start
	• Stop
		Windows:
		`EGO_CONFDIR\audits\egoservice.audit.log`
		Linux:
		`EGO_CONFDIR/audits/egoservice.audit.log`

Host	• Open
	• Close
		Windows:
		`EGO_CONFDIR\audits\ego.audit.log`
		Linux:
		`EGO_CONFDIR/audits/ego.audit.log`

User	• Add
	• Modify
	• Delete
	• Assign a new role
	• Unassign a role
	• Log on from GUI/CLI
	• Log off from GUI/CLI
	• Log on fail from GUI/CLI/API
		Windows:
		`EGO_CONFDIR\audits\ego.audit.log`
		Linux:
		`EGO_CONFDIR/audits/ego.audit.log`

Consumer	• Add
	• Modify
	• Delete
	• Change resource plan
		Windows:
		`EGO_CONFDIR\audits\ego.audit.log`
		Linux:
		`EGO_CONFDIR/audits/ego.audit.log`

Audit log file format

Both EGO audit log files present logged events in the same format. An example is provided here for each of the EGO components and corresponding events.

DATE/TIME	TYPE	USER	OBJECT	ID	ACTION	DETAIL
time_stamp	CONTROL	*user_name*	SERVICE	*service_name*	started	-
time_stamp	CONTROL	*user_name*	SERVICE	*service_name*	stopped	-
time_stamp	CONTROL	*user_name*	SERVICE	*service_name*	start_failed	*error _msg*
time_stamp	CONTROL	*user_name*	SERVICE	*service_name*	stop_failed	*error _msg*
time_stamp	CONTROL	*user_name*	HOST	*host_name*	opened	-

DATE/TIME	TYPE	USER	OBJECT	ID	ACTION	DETAIL
time_stamp	CONTROL	*user_name*	HOST	*host_name*	closed	-
time_stamp	CONTROL	*user_name*	HOST	*host_name*	open_failed	*error _msg*
time_stamp	CONTROL	*user_name*	HOST	*host_name*	close_failed	*error _msg*
time_stamp	CONFIG	*user_name*	USER	*user_name*	created	*user_info*
time_stamp	CONFIG	*user_name*	USER	*user_name*	modified	*user_info*
time_stamp	CONFIG	*user_name*	USER	*user_name*	deleted	-
time_stamp	CONFIG	*user_name*	USER	*user_name*	assigned	*details*
time_stamp	CONFIG	*user_name*	USER	*user_name*	un-assigned	*details*
time_stamp	SECURITY	*user_name*	USER	*user_name*	logon	*caller_ip*
time_stamp	SECURITY	*user_name*	USER	*user_name*	logoff	*caller_ip*
time_stamp	CONFIG	*user_name*	USER	*user_name*	create_failed	*error _msg*
time_stamp	CONFIG	*user_name*	USER	*user_name*	modify_failed	*error _msg*
time_stamp	CONFIG	*user_name*	USER	*user_name*	delete_failed	*error _msg*
time_stamp	CONFIG	*user_name*	USER	*user_name*	assign_failed	*error _msg*
time_stamp	CONFIG	*user_name*	USER	*user_name*	un-assign_failed	*error _msg*
time_stamp	SECURITY	-	USER	*who_string**	logon_fail	*caller_ip*
time_stamp	CONFIG	*user_name*	CONSUMER	*consumer_name*	added	*details*
time_stamp	CONFIG	*user_name*	CONSUMER	*consumer_name*	modified	*details*
time_stamp	CONFIG	*user_name*	CONSUMER	*consumer_name*	deleted	-
time_stamp	CONFIG	*user_name*	CONSUMER	*consumer_name*	add_failed	*error _msg*
time_stamp	CONFIG	*user_name*	CONSUMER	*consumer_name*	modify_failed	*error _msg*
time_stamp	CONFIG	*user_name*	CONSUMER	*consumer_name*	delete_failed	*error _msg*
time_stamp	CONFIG	*user_name*	CPUPLAN	*consumer_name*	modified	*details*
time_stamp	CONFIG	*user_name*	CPUPLAN	*consumer_name*	modify_failed	*error _msg*

Note:

*As the user name cannot be acquired when a logon fails, a who_string is instead logged with the format `port@ip`.

Note:

With the exception of egosh user logon and egosh user logoff, three events are logged for commands: logon, the command, logoff. This is because authentication

is required for command-line interfaces. For the command egosh user logon and egosh user logoff, only two events are logged: logon and logoff.

PART

V

Files

ego.sudoers

Contents

About ego.sudoers

The `ego.sudoers` file is an optional file to configure security mechanisms. It is not installed by default.

You use `ego.sudoers` to grant permission to users other than root to perform certain operations as root in EGO.

The parameters in this file apply to UNIX hosts only. They are not required for Windows hosts because all users with membership in the Platform services admin group can start EGO daemons.

If `ego.sudoers` does not exist, only root can perform these operations in EGO on UNIX.

The ego.sudoers file

In EGO, certain operations such as daemon startup can only be performed by root. The `ego.sudoers` file grants root privileges to specific users or user groups to perform these operations.

Location

`ego.sudoers` must be located in `/etc` on each host.

Permissions

`ego.sudoers` must have permission 600 and be readable and writable only by root.

File format

Each entry can have one of the following forms:

- `NAME=VALUE`
- `NAME=`
- `NAME="STRING1 STRING2 ..."` except for the parameter EGO_STARTUP_ALTERNATE_PATHS, which has the format `NAME=STRING1:STRING2 ...`

The equal sign = must follow each `NAME` even if no value follows and there should be no space beside the equal sign.

`NAME` describes an authorized operation.

`VALUE` is a single string or multiple strings. The value for EGO_STARTUP_USERS is separated by spaces and enclosed in quotation marks. The value for EGO_STARTUP_ALTERNATE_PATHS is separated by colons.

Example ego.sudoers file

```
EGO_STARTUP_PATH=/usr/share/ego/etc EGO_STARTUP_ALTERNATIVE_PATHS=/usr/share/ego_cluster_1/ego/
1.2/aix5-64/etc:/usr/share/ego_cluster_2/ego/1.2/aix5-64/etc EGO_STARTUP_USERS="user1 user10
user55"
```

Creating and modifying ego.sudoers

You can modify `ego.sudoers` with a text editor if you need to specify an alternate path or paths for a parallel installation.

This file enables the EGO daemon startup control feature when EGO_STARTUP_USERS is also defined. Define both parameters when you want to allow users other than root to start EGO daemons.

Parameters

- EGO_STARTUP_PATH
- EGO_STARTUP_ALTERNATE_PATHS
- EGO_STARTUP_USERS

EGO_STARTUP_PATH

Syntax

EGO_STARTUP_PATH=*path*

Description

Specifies the absolute path name of the directory in which the EGO daemon binary files (`lim`, `vemkd`, and `egosc`) are installed. EGO daemons are usually installed in the path specified by EGO_SERVERDIR defined in the `cshrc.ego` or `profile.ego` files.

Default

Not defined. Only the root user account can start EGO daemons.

EGO_STARTUP_ALTERNATE_PATHS

Syntax

EGO_STARTUP_ALTERNATE_PATHS=*path*:*path*...

Description

For parallel installations and clusters, provides alternate paths to control multiple clusters. Define both parameters when you have multiple parallel installation paths to the directories of the EGO daemon binary files (`lim`, `vemkd`, and `egosc`) and want to allow users other than root to start EGO daemons.

EGO daemons are usually installed in the path specified by EGO_SERVERDIR defined in the `cshrc.ego` or `profile.ego` files.

The maximum length of the path string is 4000 characters.

Default

Not defined. Only the root user account can start EGO daemons.

EGO_STARTUP_USERS

Syntax

EGO_STARTUP_USERS=all_admins | **"***user_name***..."**

Description

Enables the EGO daemon startup control feature when EGO_STARTUP_PATH is also defined. Define both parameters when you want to allow users other than root to start EGO daemons.

On UNIX hosts, by default only root can start EGO daemons. To manually start EGO daemons, a user runs the command `egosh`, which has been made setuid root by the egosetsudoers script. EGO_STARTUP_USERS specifies a list of user accounts that can successfully run the command `egosh` to start EGO daemons.

all_admins

- Allows all UNIX users defined as EGO administrators in the file `ego.cluster.`*`cluster_name`* to start EGO daemons as root by running the `egosh` command.
- Not recommended due to the security risk of a non-root EGO administrator adding to the list of administrators in the `ego.cluster.`*`cluster_name`* file.

"*user_name*..."

- Allows the specified user accounts to start EGO daemons by running the `egosh` command.
- Separate multiple user names with a space.
- For a single user, do not use quotation marks.

Default

Not defined. Only the root user account can start EGO daemons.

Index

A

C

D

E

F

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W