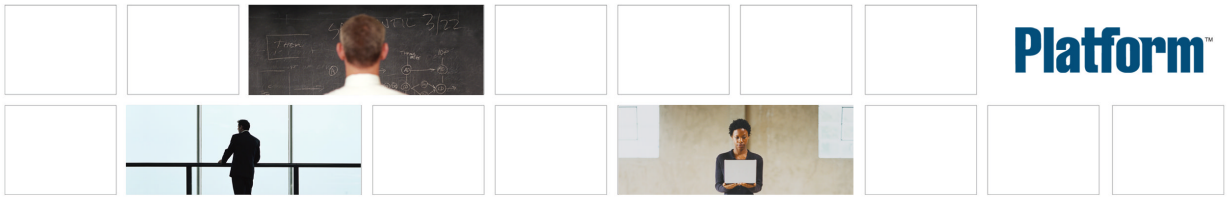
Reference

Platform Symphony™
Version 4.1
November 2008



Contents

Part I: Commands ... 5

Symphony command summary ... 7
egoconfig ... 10
egoremoverc ... 15
egosetrc ... 16
egosetsudoers ... 17
egosh ... 18
egoshutdown ... 47
egostartup ... 48
pversions ... 49
rfa ... 50
rsdeploy ... 55
soamcontrol ... 61
soamdeploy ... 70
soamlog ... 75
soamlogoff ... 81
soamlogon ... 82
soammod ... 83
soamreg ... 87
soamshutdown ... 91
soamstartup ... 92
soamswitch ... 93
soamunreg ... 97
soamview ... 99
symexec ... 106
symping ... 110

Part II: Application Profile ... 117

Application profile reference ... 119
Consumer section ... 123
SOAM section ... 137
SessionTypes section ... 162
Service section ... 171

Part III: Symphony Resources ... 197

Host properties 199
Load indices 201
-R res_req 203

Part IV: Configuration Files ... 209

deployment.xml 211
ego.conf (for Clients) 214
vem_resource.conf 216
ego.sudoers 219

Part V: Environment Variables ... 223

Symphony client environment variables 225
EGO environment variables 231

PART

Commands

Symphony command summary

Platform Symphony provides commands for various purposes.

Development environment commands

The following commands are available in Symphony DE.

Command	Description
soamcontrol	Controls applications, sessions, and tasks.
soamdeploy	Deploys, removes, and displays information about service packages for consumers.
soamlog	Dynamically changes the log level for Symphony components.
soammod	Modifies session priority to enable high priority workload to finish faster.
soamreg	Registers an application or updates the application profile of a registered application.
soamunreg	Unregisters an application and deletes the application profile preventing more sessions from being started for this application.
soamstartup	In Symphony DE only, starts Symphony processes on the local host.
soamshutdown	In Symphony DE only, immediately shuts down Symphony processes on the local host.
soamswitch	Windows only. Resets the system environment for the application client: connecting to a DE cluster from the DE installation environment; connecting to a Symphony cluster from the DE installation environment; or connecting to a Symphony cluster using the Symphony installation environment.
soamview	Displays information about applications, sessions, and tasks, and displays the application profile.
symexec	Run executables as Symphony applications.
symping	Sends workload to a cluster to test and verify that Symphony components are working and responsive.

Cluster management and control commands

The following commands are available in Symphony grid.

Command	Description
egosh	Launches the administrative command interface to EGO.
egostartup	(script) Starts all EGO components of a cluster.
egoshutdown	(script) Shuts down a cluster.

Command	Description
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.
rfa	Transfers files between hosts.
rsdeploy	Deploys and removes middleware packages. You must be a cluster administrator to run this command.
soamlog	Dynamically changes the log level for Symphony components.

Cluster configuration commands

The following commands are available in Symphony grid.

Command	Description
egoconfig	Configures hosts.
egosetrc	Configures automatic startup of EGO on a UNIX host.
egoremoverc	Prevents automatic startup of EGO on a UNIX host.
egosetsudoers	Creates an etc/ego/sudoers file to determine accounts with root privileges on the UNIX host within a cluster.

Workload management commands

The following commands are available in Symphony grid.

Command	Description
soamcontrol	Controls applications, sessions, and tasks.
soamdeploy	Deploys, removes, and displays information about service packages for consumers.
soamlogon	Logs the user on to Platform Symphony for a specific time period.
soamlogoff	Ends the login session with Platform Symphony.
soammod	Modifies session priority to enable high priority workload to finish faster.
soamreg	Registers an application or updates the application profile of a registered application.
soamunreg	Unregisters an application and deletes the application profile preventing more sessions from being started for this application.
soamswitch	Windows only. Resets the system environment for the application client: connecting to a DE cluster from the DE installation environment; connecting to a Symphony cluster from the DE installation environment; or connecting to a Symphony cluster using the Symphony installation environment.
soamview	Displays information about applications, sessions, and tasks, and displays the application profile.

Command	Description
symexec	Run executables as Symphony applications.
symping	Sends workload to a cluster to test and verify that Symphony components are working and responsive.

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. For most subcommands, 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 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]

Log on using the installation OS account, or with the same permissions as the account you used to install on the host. For example, if you installed without Windows system administrator permissions, log on as cluster administrator. If you installed with Windows system administrator permissions on the local host, log on using any account that has Windows system administrator permissions on the local host.

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 cluster administrator OS user account. The format is `DOMAIN \user_name`.

password

Specifies the password to use to authenticate the user account (input the actual password of the cluster administrator OS account).

-f

Suppresses confirmation of the command.

Use this option when running this command from a script.

mghost soam [-f]

For use on UNIX. Use this command when adding Platform Symphony to an existing cluster. Specifies to ignore the local Symphony configuration directory and use the EGO shared directory instead. Issue this command only on the master host so that Symphony 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 soam [*user_account*] [*password]* [-f]

Log on using the installation OS account, or with the same permissions as the account you used to install on the host. For example, if you installed without Windows system administrator permissions, log on as cluster administrator. If you installed with Windows system administrator permissions on the local host, log on using any account that has Windows system administrator permissions on the local host.

For use on Windows. Use this command when adding Platform Symphony to an existing cluster. Specifies to ignore the local Symphony configuration directory and use the EGO shared directory instead. Issue this command only on the master host so that Symphony can use shared location configurations.

user_account

Specifies the cluster administrator OS user account. The format is `DOMAIN\user_name`.

password

Specifies the password to use to authenticate the user account (input the actual password of the cluster administrator OS 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.

The following port numbers are also changed as a result of issuing this command:

- EGO_LIM_PORT
- EGO_KD_PORT
- EGO_PEM_PORT
- ESC_PORT
- REPOSITORY_SERVICE_PORT
- SD_ADMIN_PORT
- SD_SDK_PORT

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.

Symphony always uses seven consecutive ports starting from the base port. By default, Symphony uses ports 7869-7875.

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]

Log on using the installation OS account, or with the same permissions as the account you used to install on the host. For example, if you installed without Windows system administrator permissions, log on as cluster administrator. If you installed with Windows system administrator permissions on the local host, log on using any account that has Windows system administrator permissions on the local host.

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 the file ego under the system startup directory.

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

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

Used 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*]…]

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 setpriority -lowest | -normal *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]

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, whether the host is currently enabled for scavenging, and whether the grace period is disabled.

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 with grace period (on), enabled without grace period (fastrelease), 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 fastrelease,2,0.3,1.67 all
```

This example enables (turns “on”) host scavenging on all hosts, disables the grace period, 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 setpriority -lowest | -normal *resource_name* …

Sets the OS process priority of cluster workloads running on scavenge-capable hosts.

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

-lowest

Specifies to set the process priority to lowest. EGO allocates this resource to run workload at the lowest process priority as controlled by the operating system.

-normal

Specifies to set the process priority to normal (default). EGO allocates this resource to run workload at normal process priority as controlled by the operating system.

resource _name ...

Specifies the name of the resource or resources on which to set the OS process priority.

Separate multiple 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 descendants:

```
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 descendant 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 descendants 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.

If you are using a firewall, you need to set the environment variable EGO_CLIENT_ADDR on the client to configure the port range for the firewall. It is required to set this variable on the client machine before running the rfa command.

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== \"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.

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== \"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.

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 packages, such as middleware, upgrade, file, or binary 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 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:

File package 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 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 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 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 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.

soamcontrol

Controls applications and sessions.

Synopsis

soamcontrol *subcommand* [*options*]

soamcontrol -h

soamcontrol *subcommand* **-h**

soamcontrol -V

Description

Use the `soamcontrol` command with a subcommand to control applications, sessions, and tasks.

Symphony DE does not verify the optional user name and password. Symphony DE users can control all sessions, applications, and tasks in their environment.

In grid, the user must have the appropriate permissions to control an application, task, or session, as follows:

- Consumer administrators have permission to use this command to control applications, sessions, and tasks in the consumers they manage
- Cluster administrators have permission to use this command to control applications, sessions, and tasks for all consumers
- Consumer users can use this command only to control sessions they submit

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Subcommand synopsis

app enable *application_name* [**-u** *user_name*] [**-x** *password*]

app disable *application_name* | **all** [**-f**] [**-s**] [**-u** *user_name*] [**-x password**]

session kill *application_name* [**:***session_ID*[**,***session_ID*]...] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

session kill *application_name* [**-t** *session_tag*] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

session suspend *application_name* [**:***session_ID*[**,***session_ID*]...] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

session suspend *application_name* [**-t** *session_tag*] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

session resume *application_name* [**:***session_ID*[**,***session_ID*]...] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

session resume *application_name* [**-t** *session_tag*] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

task kill *application_name***:***session_ID*:*task_ID*[,*task_ID*...] [**-m** *comment*] [**-u** *user_name*] [**-x** *password*]

app enable *application_name* [-u *user_name*] [-x *password*]

Enables an application for a consumer with no other enabled applications.

There can only be one enabled application per consumer.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only, the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you have already logged on to Symphony using `soamlogon`, for this command only the password you specify here overrides the password you entered in `soamlogon`.

Enable sampleApp application.

soamcontrol app enable sampleApp

app disable *application_name* | all [-f] [-s] [-u *user_name*] [-x *password*]

Disables a specific application or with the all option, disables all applications to which the specified user has access.

When issued by a cluster administrator, can disable all applications under all consumers in the cluster.

When issued by a consumer administrator, can disable all applications under the consumer this administrator administers.

When an application that has running workload is disabled, all active sessions are immediately killed and all system resources assigned to the application are released.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

all

Disables all applications and kills all workload.

-f

Disables the application and kills application workload without warning if there is active workload for the application.

-s

Disables the application and saves recoverable application workload. Prompts for confirmation to disable the workload. Saved workload is recovered when the application is enabled again.

-u user_name

Specifies the name of the user to connect to Platform Symphony for this command. If you have already logged on to Platform Symphony using `soamlogon`, for this command only the user name you specify here overrides the user name you entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Disable all applications

Disable all applications to which the consumer administrator has access.

soamcontrol app disable all -u consumer_admi -x passwd

Disable application and shutdown workload

Disable application sampleApp and shut down any existing workload.

soamcontrol app disable sampleApp -f

session kill *application_name* [:*session_ID*[,*session_ID*]...] [-m *comment*] [-u *user_name*] [-x *password*]

Terminates all sessions of an application or terminates a specific session of an application. The session runtime data and its tasks (such as output data and current state) are no longer available. However, historical data can be retrieved through the `soamview` or the Platform Symphony GUI if session and task history is saved.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

session_ID

Specifies the Platform Symphony -assigned ID of the session as reported by `soamview session`.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user_name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Kill all sessions

Kill all sessions of the sampleApp application.

soamcontrol session kill sampleApp

Kill specific sessions

Kill sessions 101 and 102 of the sampleApp application.

soamcontrol session kill sampleApp:101,102

session kill *application_name* [-t *session_tag*] [-m *comment*] [-u *user_name*] [-x *password*]

Terminates all sessions of an application or terminates sessions that match the specified session tag of an application. The session run-time data and its tasks (such as output data and current state) are no longer available. However, historical data can be retrieved through the `soamview` or the Platform Symphony GUI if session and task history is saved.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-t session_tag

Specifies the string that is used for identification purposes to control the session.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user_name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Kill all sessions that match the session tag

Kill all sessions of the sampleApp application that share the "myTag" session tag.

soamcontrol session kill sampleApp -t myTag

session suspend *application_name* [:*session_ID*[,*session_ID*]...] [-m *comment*] [-u *user_name*] [-x *password*]

Suspends a session. When a session is suspended, running tasks are killed and rescheduled for rerun when the session is resumed. Pending tasks remain pending. CPU slots used by the session are freed.

Clients can continue to create sessions, send tasks, retrieve task output, and query session and task information.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

If specified without one or more session IDs, all sessions of the application are suspended.

session_ID

Specifies the Symphony-assigned ID of the session as reported by `soamview session`.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user_name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Suspend all sessions

Suspend all sessions of the sampleApp application.

soamcontrol session suspend sampleApp

Suspend specific sessions

Suspend sessions 101 and 102 of the sampleApp application, giving a reason for the suspension.

soamcontrol session suspend sampleApp:101,102 -m "network congestion"

session suspend *application_name* [-t *session_tag*] [-m *comment*] [-u *user_name*] [-x *password*]

Suspends all sessions of an application or suspends sessions that match the specified session tag of an application. When a session is suspended, running tasks are killed and rescheduled for rerun when the session is resumed. Pending tasks remain pending. CPU slots used by the session are freed.

Clients can continue to create sessions, send tasks, retrieve task output, and query session and task information.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

If specified without one or more session IDs, all sessions of the application are suspended.

-t session_tag

Specifies the string that is used for identification purposes to control the session.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Suspend all sessions that match the session tag

Suspend all sessions of the sampleApp application that share the "myTag" session tag.

soamcontrol session suspend sampleApp -t myTag -m "network congestion"

session resume *application_name* [:*session_ID*[,*session_ID*...]] [-m *comment*] [-u *user_name*] [-x *password*]

Resumes task processing for a specific suspended session or for multiple sessions.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

If specified without one or more session IDs, all sessions of the application are resumed.

session_ID

Specifies the Platform Symphony -assigned ID of the session as reported by `soamview session`.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Resume all sessions

Resume all sessions of the sampleApp application.

soamcontrol session resume sampleApp

Resume a specific session

Resume sessions 101 and 102 of the sampleApp application, giving a reason for the resumption.

soamcontrol session resume sampleApp:101,102 -m "network problem resolved"

session resume *application_name* [-t *session_tag*] [-m *comment*] [-u *user_name*] [-x *password*]

Resumes all sessions of an application or resumes sessions that match the specified session tag of an application.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

If specified without one or more session IDs, all sessions of the application are resumed.

-t session_tag

Specifies the string that is used for identification purposes to control the session.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Resume sessions that match the session tag

Resume all sessions of the sampleApp application that match the "myTag" session tag.

soamcontrol session resume sampleApp -t myTag -m "network problem resolved"

task kill *application_name*:*session_ID*:*task_ID*[,*task_ID*...] [-m *comment*] [-u *user_name*] [-x *password*]

Kills one or more specific tasks in a running or pending session.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

session_ID

Specifies the Platform Symphony -assigned ID of the session as reported by `soamview session`.

task_ID

Specifies the Platform Symphony -assigned ID of the task as reported by `soamview task`.

-m comment

Specifies a reason why this action was requested. The reason can comprise up to 256 alphanumeric or special characters, except greater than (>), less than (<), ampersand (&), single quote (') on Windows, or control characters (Ctrl + key). The description must be enclosed in double quotes (" ") if it contains spaces.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Kill specified tasks in a session

Kill tasks 201 and 202 of the sampleApp application in session 101.

task kill sampleApp:101:201,202 -m "tasks hanging"

soamdeploy

Deploys, removes, and displays information about service packages for consumers.

Synopsis

soamdeploy

soamdeploy *subcommand* [*options*]

soamdeploy *subcommand* **-h**

soamdeploy -h

soamdeploy -V

Description

Use the `soamdeploy` command to deploy, verify, and delete service packages

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Subcommand synopsis

add package_name -p *package_file* **-c** *consumer_ID* [**-f**] [**-n**] [**-u** *user_name*] [**-x** *password*]

remove package_name -c *consumer_ID* [**-f**] [**-u** *user_name*] [**-x** *password*]

view -c *consumer_ID* [**-u** *user_name*] [**-x** *password*]

add package_name -p *package_file* -c *consumer_ID* [-f] [-n] [-u *user_name*] [-x *password*]

Copies the specified package to the central repository.

When you add a package with a new package name to the repository, the package file is stored in the repository. When you add a package with an existing package name:

Note:

If a package is shared among multiple applications, then the following points are applicable for each application.

- The package in the repository is replaced.
- If the package is being used by an enabled application, workload continues to run with the next scheduled task using the updated service package.

When a new session requests the service in the package, the package is deployed and uncompressed on to compute hosts on-demand.

package_name

Name you want to assign to the service package. The service package name must be unique within a consumer, with a maximum of 1024 characters. Valid characters are alphanumeric characters, periods, and underscores. Spaces are not allowed.

-p package_file

Path and file of the service package to deploy.

-c consumer_ID

Consumer for which to deploy the service packages. Only applications registered to this consumer and below in the consumer tree can use the service packages.

Enclose the consumer ID in double quotes (" ") if it contains spaces.

If a service package is deployed to the root consumer ("/"), the package is shared by the leaf consumers.

Note:

For Symphony DE, -a application profile file name can be used instead of -c consumer_ID.

-f

Forces package deployment without prompting. Note that if you use this option and the package was previously deployed, any running workload is terminated without prompting. Use this option when you are issuing `soamdeploy add` from within a script, and do not want the script to stop running to respond to prompts.

-n

Turns off package verification during deployment. Use this option if you are deploying the same package repeatedly and you know it is valid, and deployment takes a long time because of the verification process.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Examples: Windows

Deploy a service package

Deploy the service package for consumer /SampleApplications/SOASamples.

```
soamdeploy add Mypackage -p sampleService.zip -c /
SampleApplications/SOASamples
```

Update a service package

Update and replace the service package `sampleService.zip`.

```
soamdeploy add Mypackage -p sampleService.zip -c /
SampleApplications/SOASamples
```

Examples: Linux

Deploy a service package

Deploy the service package for consumer /SampleApplications/SOASamples.

```
soamdeploy add Mypackage -p sampleService.tar.gz -c /
SampleApplications/SOASamples
```

Update a service package

Update and replace the service package

```
soamdeploy add Mypackage -p sampleService.tar.gz -c /
SampleApplications/SOASamples
```

remove *package_name* -c *consumer_ID* [-f] [-u *user_name*] [-x *password*]

Deletes the specified service package from the central repository under the specified consumer.

Attention:

The package is removed only from the central repository and is not removed from the remote deploy directory until another package is uploaded to the same consumer on that host.

You cannot remove a package if there are registered applications using the package. Unregister the application(s) with `soamunreg` before attempting to remove the package.

Note:

If your application no longer needs to reference the service package and you do not want to unregister it, you can remove reference to the package in the application profile and update the application profile with the `soamreg` command.

package_name

package_name

Name you assigned to the service package during deployment.

-c consumer_ID

Consumer from which to remove the service package.

Enclose the consumer ID in double quotes (" ") if it contains spaces.

-f

Forces package removal without prompting. Use this option when you are issuing `soamdeploy remove` from within a script, and do not want the script to stop running to respond to prompts.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Remove a service package

Remove the service package for consumer /SampleApplications/SOASamples.

```
soamdeploy remove Mypackage -c /SampleApplications/
SOASamples
```

view -c *consumer_ID* [-u *user_name*] [-x *password*]

Lists all deployed service packages for the specified consumer.

-c consumer_ID

Displays a list of all deployed service packages for the specified consumer. Enclose the consumer ID in double quotes (" ") if it contains spaces.

Note:

For Symphony DE, -c consumer_ID is optional.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

View deployed packages for consumer /SampleApplications/SOASamples

```
soamdeploy view -c /SampleApplications/SOASamples
```

soamlog

Dynamically changes the log level for Symphony components.

Synopsis

soamlog *subcommand* [*options*]

soamlog -h

soamlog -V

Description

Use `soamlog` to dynamically change the level of detail logged for messages and errors. The log-level change becomes effective immediately and remains in effect until the host is restarted or the affected process or application is closed.

Symphony component log files are written to the `logs` directory in SOAM_HOME on the host where the process runs.

When the client path is in the current user's path environment, application logs are written to the directory where the client executable is run. If the client executable path is not in the current user's path environment, application logs are written to the directory where the client executable resides.

Use the `log4j.properties` file on the host on which a Symphony component runs to customize the contents of the log files for that component. The `log4j.properties` file is located in the conf directory in SOAM_HOME.

In Symphony DE, all users can change any log level.

In Symphony, the cluster or consumer administrator must have permission to use this command. The cluster administrator can change the logging options for the Session Director (sd). The consumer administrator can change the log levels for the session manager (ssm), service instance manager (sim) and workload of their applications.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Subcommand synopsis

sd_all | *sd_logger* **-l** *log_level* **[-u** *user_name* **] [-x** *password* **]**

ssm_all | *ssm_logger application_name* **-l** *log_level* **[-u** *user_name* **] [-x** *password* **]**

sim_all | *sim_logger application_name* **-l** *log_level* **[-u** *user_name* **] [-x** *password* **]**

soamdeploy_all | *soamdeploy_logger application_name* **-l** *log_level* **[-u** *user_name* **] [-x** *password* **]**

workload *application_name* **-l** *log_level* **[-u** *user_name* **] [-x** *password* **]**

sd_all | *sd_logger* -l *log_level* [-u *user_name*] [-x *password*]

sd_all

For activity relating to the Session Director, changes the log level for all logger classes in the `sd.log4j.properties` file, including those without the sd prefix.

sd_logger

Specifies the name of the log4j.logger you want to change the log level of. For example:

- SD—Changes the log level for all logger classes in `sd.log4j.properties`, that are prefixed with `sd`.

Refer to the appropriate `log4j.properties` file (in `$SOAM_HOME/conf` on Linux, `%SOAM_HOME%\conf` on Windows) for the names of all related loggers. If an invalid logger name is specified, the component log levels are not changed.

-l log_level

Specifies one of the following log levels (listed in order of increasing detail):

Level	Description
LOG_INFO	Log all informational messages and more serious messages.
LOG_WARN	Log only those messages that are warnings or more serious messages. This is the default level of debug information.
LOG_ERROR	Log only those messages that indicate error conditions.
LOG_FATAL	Log only those messages in which the system is unusable.
LOG_DEBUG	Log all debug-level messages.
LOG_ALL	Log all available messages.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the log level of the session director

Change the log level of all session director loggers to LOG_ERROR.

soamlog sd_all -l LOG_ERROR

ssm_all | *ssm_logger application_name* -l *log_level* [-u *user_name*] [-x *password*]

ssm_all

For activity relating to the specified application and the related session manager, changes the log level of all logger classes in the `ssm.log4j.properties` file, including those without the `ssm` prefix.

ssm_logger

Specifies the name of the log4j.logger you want to change the log level of. For example:

- ssm—Changes the log level of all logger classes in the `ssm.log4j.properties` file, that are prefixed with `ssm`.
- ssm.ssmbackend—Changes the log level of all logger classes in the `ssm.log4j.properties` file, that are prefixed with `backend`.

Refer to the appropriate `log4j.properties` file (in`$SOAM_HOME/conf` on Linux, `%SOAM_HOME%\conf` on Windows) for the names of all related loggers. If an invalid logger name is specified, the component log levels are not changed.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

This application must be registered. Log levels of enabled applications can be changed.

-l log_level

Specifies one of the following log levels (listed in order of increasing detail):

Level	Description
LOG_INFO	Log all informational messages and more serious messages.
LOG_WARN	Log only those messages that are warnings or more serious messages. This is the default level of debug information.
LOG_ERROR	Log only those messages that indicate error conditions.
LOG_FATAL	Log only those messages in which the system is unusable.
LOG_DEBUG	Log all debug-level messages.
LOG_ALL	Log all available messages.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the log level of the session manager backend loggers

Change the log level of the ssm backend loggers for the sampleApp application to LOG_FATAL.

soamlog ssm.ssmbackend sampleApp -l LOG_FATAL

sim_all | *sim_logger application_name* -l *log_level* [-u *user_name*] [-x *password*]

sim_all

For activity relating to the specified application and the service instance manager, changes the log level for all logger classes in the `sim.log4j.properties` file, including those without the `sim` prefix.

sim_logger

Specifies the name of the `log4j.logger` for which you want to change the log level. For example:

- sim—Changes the log level for all logger classes in the `sim.log4j.properties` file, that are prefixed with `sim`.
- sim.backend—Changes the log level for all logger classes in the `sim.log4j.properties` file, that are prefixed with `backend`.

Refer to the appropriate `log4j.properties` file (in `$SOAM_HOME/conf` on Linux, `%SOAM_HOME%\conf` on Windows) for the names of all related loggers. If you specify an invalid logger name, `soamlog` does not change the log level for any component.

If you specify an invalid logger name, `soamlog` does not change the log level for any component.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-l log_level

Specifies one of the following log levels (listed in order of increasing detail):

Level	Description
LOG_INFO	Log all informational messages and more serious messages.
LOG_WARN	Log only those messages that are warnings or more serious messages. This is the default level of debug information.
LOG_ERROR	Log only those messages that indicate error conditions.
LOG_FATAL	Log only those messages in which the system is unusable.
LOG_DEBUG	Log all debug-level messages.
LOG_ALL	Log all available messages.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the log level of the service instance manager for an application

Change the log level of the sim backend loggers for the sampleApp application to LOG_WARN.

soamlog sim.backend sampleApp -l LOG_WARN

soamdeploy_all | *soamdeploy_logger application_name* -l *log_level* [-u *user_name*] [-x *password*]

soamdeploy_all

For activity relating to soamdeploy that are started by SIM, changes the log level for all logger classes in the `soamdeploy.log4j.properties` file, including those without the `soamdeploy` prefix.

soamdeploy_logger

Specifies the name of the `log4j.logger` for which you want to change the log level. For example:

- soamdeploy—Changes the log level for all logger classes in the `soamdeploy.log4j.properties` file, that are prefixed with `soamdeploy`.

Refer to the appropriate `log4j.properties` file (in `$SOAM_HOME/conf` on Linux, `%SOAM_HOME%\conf` on Windows) for the names of all related loggers. If you specify an invalid logger name, `soamlog` does not change the log level for any component.

If you specify an invalid logger name, `soamlog` does not change the log level for any component.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-l log_level

Specifies one of the following log levels (listed in order of increasing detail):

Level	Description
LOG_INFO	Log all informational messages and more serious messages.
LOG_WARN	Log only those messages that are warnings or more serious messages. This is the default level of debug information.
LOG_ERROR	Log only those messages that indicate error conditions.

Level	Description
LOG_FATAL	Log only those messages in which the system is unusable.
LOG_DEBUG	Log all debug-level messages.
LOG_ALL	Log all available messages.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the log level of soamdeploy for an application

Change the log level of all soamdeploy loggers to LOG_ERROR.

soamlog soamdeploy_all sampleApp -l LOG_ERROR

workload *application_name* -l *log_level* [-u *user_name*] [-x *password*]

For activity relating to the specified application, changes the log level of workload-related log entries in the session manager (ssm), session director (sd) and service instance manager (sim) log files.

Change the workload log level for an application

Change the log level of the sampleApp application

soamlog workload sampleApp -l LOG_DEBUG

soamlogoff

Ends the login session with Platform Symphony .

Synopsis

soamlogoff

soamlogoff -h

soamlogoff -V

Description

In Symphony, use `soamlogoff` to end a Symphony user session.

It is not necessary to log on or log off from Symphony DE.

Once logged off, the user must log on again to issue Symphony commands or must enter a name and password as options to the Symphony command.

It is not necessary to log off to issue commands under a different user account. Specifying a user name and password on the command line overrides the user name and password specified with `soamlogon`.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

soamlogon

Logs the user on to Platform Symphony for a specific time period.

Synopsis

soamlogon [**-u** *user_name*] [**-x** *password*] [**-t** *time-to-live*]

soamlogon -h

soamlogon -V

Description

Use `soamlogon` to log on to Symphony for a specific time period.

You do not need to log on to use Symphony DE.

When `soamlogon` is used without any options, the system prompts for a user name and password. Symphony does not prompt for an expiry time; it uses the default command session length, which is 8 hours.

Once the command session time-to-live expires, the command session is terminated.

The user name and password specified using `soamlogon` apply to all commands issued during this command session, unless they are overridden on the command line using the user name (**-u**) and password (**-x**) command options.

When a command is run with the **-u** and **-x** options, the system checks the user name and password to ensure the user has the required authority to run that command. The user name and password entered as options to a command apply only to that command instance.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Options

-t time-to-live

Default=480 minutes (8 hours). Specifies the command session duration in minutes. When the time expires, you are automatically logged off the system. To run more commands, you must log on again or provide the user name and password on the command line as command options.

-u user _name

Specifies the name of the user to connect to Symphony for this command.

-x password

Specifies the user password to connect to Symphony for this command.

soammod

Modifies session priority to enable high priority workload to finish faster.

Synopsis

soammod *subcommand* [options]

soammod session -h

soammod session -V

Description

Use `soammod session` to change the priority of one session, all sessions, or sessions that match the session tag of an application. Symphony increases or decreases the amount of resources assigned to a session depending on session priority.

Any priority changes made are applied at a time determined by the Consumer > sessionSchedulingInterval attribute of the application profile. When Symphony next checks the workload to determine if it must reassign resources, it examines any session priority changes and reallocates resources accordingly.

Changing the priority of a session does not affect running tasks. The system waits for the task to finish before reassigning the resource.

Platform Symphony DE does not verify the optional user name and password. Platform Symphony DE users can control all sessions and applications in their environment.

In the Symphony environment:

- Consumer users can only change sessions they submitted.
- Consumer administrators can change sessions of all applications under that consumer.
- Cluster administrators can change all sessions under all consumers in the cluster.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Subcommand synopsis

session *application_name* [**-u** *user_name*] [**-x** *password*] **-p** *priority_number*

session *application_name*:*session_ID* [**-u** *user_name*] [**-x** *password*] **-p** *priority_number*

session *application_name* [**-u** *user_name*] [**-x** *password*] **-p** *priority_number* **-t** *session_tag* [**-f**]

session *application_name* [**-u** *user_name*] [**-x** *password*] **-r "priority" -t** *session_tag*

session *application_name* [-u *user_name*] [-x *password*] -p *priority_number*

Changes the priority of all sessions for the application.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it

contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-p priority_number

Specifies the priority number to assign to all sessions of the specified application. The priority number range is from 1 to 10,000, where priority 1 is the lowest priority.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the priority of all sessions

Change the priority of all sampleApp sessions to 10.

soammod session sampleApp -p 10

session *application_name*:*session_ID* [-u *user_name*] [-x *password*] -p *priority_number*

Changes the priority of a specific session for the application.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

session_ID

Specifies the Platform Symphony-assigned ID of the session as reported by `soamview session`.

-p priority_number

Specifies the priority number to assign to the specified session of the specified application. The priority number range is from 1 to 10,000, where priority 1 is the lowest priority.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the priority of a specific session.

Change the priority of the specific session to 100.

soammod session sampleApp:101 -p 100

session *application_name* [-u *user_name*] [-x *password*] -p *priority_number* -t *session_tag* [-f]

Changes the priority of all sessions for the application that share the specified session tag.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-p priority_number

Specifies the priority number to assign to all current sessions of the specified application that share the given session tag. The priority number range is from 1 to 10,000, where priority 1 is the lowest priority.

-t session_tag

Specifies the string that is used for identification purposes to control the session.

-f

Specifies that future sessions created with the given session tag will inherit the specified priority, rather than get their priority from the application profile. The -f flag can only be used when the -t session_tag option and the -p priority option are both specified. If the -f flag is not specified, the priority change applies only to current sessions that share the given session tag.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Change the priority of current and future sessions

Change the priority of all current and future sampleApp sessions that share the "myTag" session tag to 10.

soammod session sampleApp -p 10 -t myTag -f

session *application_name* [-u *user_name*] [-x *password*] -r *"priority*" -t *session_tag*

Resets the priority of all future sessions for the application that share the specified session tag.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-r "priority"

Resets the priority change that was previously specified with the -f flag for future sessions. Future sessions that are created with the matching session tag will use the priority from the application profile.

-t session_tag

Specifies the string that is used for identification purposes to control the session.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Reset the priority of future sessions

Reset the priority of future sampleApp sessions created with the "myTag" session tag. Future sessions created with the "myTag" session tag will get their priority from the application profile.

soammod session sampleApp -t myTag -r "priority"

soamreg

Registers an application or updates the application profile of a registered application.

Synopsis

soamreg *application_profile_file* [**-d**][**-f**] [**-u** *user_name*] [**-x** *password*]

soamreg *fragment_profile_file* **-p** [**-f**] [**-u** *user_name*] [**-x** *password*]

soamreg -h

soamreg -V

Description

Use `soamreg` to register a new application or update an application with an existing profile or profile fragment.

Before you register an application in Symphony:

- The consumer specified in the application profile must already exist in the resource group names specified, or in the default resource groups.
- The service packages specified in the application profile or profile fragment must have already been deployed for the consumer

Remember:

If the application profile resourceGroupName attribute for the session manager and service instance manager is not defined, the default compute and management host groups are used. In this case, the consumer must be defined in the `ComputeHost` and the `ManagementHost`. Otherwise, the application cannot be enabled after registration.

In Symphony DE, the consumer name is optional and not used.

The table that follows describes Symphony behavior when registering an application, based on existing conditions when `soamreg` was issued:

	No enabled application in consumer	No enabled application in consumer	Enabled application in consumer	Enabled application in consumer	Enabled application in consumer	Enabled application in consumer
			Application *has* open or suspended sessions	Application *has* open or suspended sessions	Application *has no* open or suspended sessions	Application *has no* open or suspended sessions
System action resulting from issuing `soamreg`	**Register new application**	**Update existing application**	**Register new application**	**Update existing application**	**Register new application**	**Update existing application**
Registers and enables the application	✓			✓		✓
Registers the application, and it remains disabled		✓	✓		✓	

	No enabled application in consumer	No enabled application in consumer	Enabled application in consumer	Enabled application in consumer	Enabled application in consumer	Enabled application in consumer
			Application *has* open or suspended sessions	Application *has* open or suspended sessions	Application *has no* open or suspended sessions	Application *has no* open or suspended sessions
System action resulting from issuing `soamreg`	**Register new application**	**Update existing application**	**Register new application**	**Update existing application**	**Register new application**	**Update existing application**
Registers the application profile fragment		✓		✓		✓
Terminates all sessions				✓		
Shuts down session manager, releases resources, and starts up a new session manager				✓		✓
Updates the application profile		✓		✓		✓
Dynamic update for application profile		✓		✓		✓

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Options

application_profile_file

Specify the path and name of the application profile or profile fragment file for the application. If any part of the path or file name contains spaces, enclose it in double quotes (" "). Specify the path for the application profile using absolute, relative, or UNC format.

You can specify file paths using absolute or relative paths:

- Absolute—Provide the full path to the file, from the root directory. If you specify an absolute path, all hosts must have the same path. For example:
  - Windows: **"%SOAM_HOME%\4.1\samples\CPP\SampleApp\SampleApp.xml"**
  - Linux: **"$SOAM_HOME/4.1\samples/CPP/SampleApp/SampleApp.xml"**
- Relative—Provide the path to the file relative to the Symphony work directory, `%SOAM_HOME%\work` in Windows and `$SOAM_HOME/work` in Linux. For example:
  - Windows: **"..\4.1\samples\CPP\SampleApp\SampleApp.xml"**

- Linux: **"../4.1/samples/CPP/SampleApp/SampleApp.xml"**

fragments_profile_file

Register a profile fragment file that contains either a new session type or a new service section, or both. When you *add* a new session type or Service section, the system does not terminate running workload. When you *update* an existing Service section or session type, the system only terminates running workload associated with the updated service or session type. The add, update, and remove actions are also applicable for multiple session types and services.

You can specify file paths using absolute or relative paths as explained in application_profile_file.

-f

Registers a new application without warning. For existing applications, updates the existing application profile and terminates application workload without warning, if there is active workload for the application.

Registers a profile fragment and does not prompt you to confirm the addition or update.

-p

Registers a profile fragment with the new session types and Service sections, or with the updated or removed session types and Service sections. You can refer to the profile fragment templates to create your own profiles. You can find templates in the `$SOAM_HOME/4.1/samples/Templates` directory.

-d

Updates the application profile dynamically. If you use this option with **-f**, then affected workload will be terminated without any warning.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Register an application

Register an application on Linux:

```
soamreg soam/4.1/samples/CPP/SampleApp/SampleApp.xml
```

Register a profile fragment:

```
soamreg soam/4.1/samples/CPP/SampleApp/
SampleAppFragment.xml -p
```

Register an application on Windows:

```
soamreg "\\hostA\share\SampleApp.xml"
```

Register a profile fragment:

```
soamreg "\\hostA\share\SampleAppFragment.xml" -p
```

soamshutdown

In Symphony DE only, immediately shuts down Symphony processes on the local host.

Synopsis

soamshutdown [**-all**]

soamshutdown -h

soamshutdown -V

Description

In Symphony DE only, use `soamshutdown` to shut down Symphony processes on the local host or all hosts in the cluster.

The taskCleanupPeriod and suspendGracePeriod for `SessionTypes` you specify in the application profile do not apply when you issue this command. The `soamshutdown` command immediately shuts down all Platform Symphony processes.

On a management host, `soamshutdown` stops the following processes:

- Session director (`sd`)
- Session manager (`ssm`)
- Repository service (`rs`)
- Platform Management Console (`webgui`) and Java processes
- Platform EGO emulator (`start_agent`)

On a compute host, `soamshutdown` stops the following processes:

- Service instance manager (`sim`)
- Platform EGO emulator (`start_agent`)

Options

-all

Immediately shuts down Symphony processes on all hosts in the cluster as configured in `vem_resource.conf`.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

soamstartup

In Symphony DE only, starts Symphony processes on the local host.

Synopsis

soamstartup

soamstartup -h

soamstartup -V

Description

In Symphony DE, use `soamstartup` to start Symphony software processes on the local host under the authority of the login user.

The processes `soamstartup` starts depend on the host role, management or compute.

- If the host is a management host, as specified in`vem_resource.conf`, `soamstartup` starts:
  - Platform EGO emulator (`start_agent`)
  - Session director (`sd`)
  - Repository service (`rs`)
  - Platform Management Console (`webgui`) and Java processes
- If the host is a compute host, `soamstartup` starts:
  - Platform EGO emulator (`start_agent`)
  - For any applications specified to be prestarted in their application profiles, `soamstartup` also starts:
    - Session manager (`ssm`)
    - Service instance manager (`sim`)
- Processes started when the session director receives workload requests or enabled applications are specified to prestart in their application profiles are:
  - Session manager (`ssm`)
  - Service instance manager (`sim`)

Attention:

For Windows users only—If Symphony DE is installed by the administrator, `start_agent` runs as a Windows service. Closing the command shell under which Symphony DE was started or logging off Windows does not stop the `start_agent` service. However, installing Symphony DE as a non-administrative user makes Symphony DE run as a normal Windows process. In this case, if you close the command shell from which `soamstartup` was issued, the start_agent process is killed but any processes it started, such as the session director or session manager continue to run and must be shut down using `soamshutdown`. If the user logs off Windows, all DE processes will be stopped.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

soamswitch

Windows only. Resets the system environment for the application client: connecting to a DE cluster from the DE installation environment; connecting to a Symphony cluster from the DE installation environment; or connecting to a Symphony cluster using the Symphony installation environment.

Synopsis

soamswitch sym

soamswitch symde

soamswitch sym *Symphony_dir* [**-v** *version*] [**-h**]

soamswitch sym *SymphonyDE_dir* [**-v** *version*] [**-h**]

soamswitch symde *SymphonyDE_dir* [**-v** *version*] [**-h**]

soamswitch info

soamswitch -h |**-V**

Description

You must have local administrator privileges on the host to use `soamswitch`.

Resets the system environment for this machine, specifying both the installation environment to use, and the cluster to connect to.

After the `soamswitch` command is issued, environment changes take effect on all subsequent command sessions—the environment in the current command session is not affected.

Options

soamswitch sym

Use this command to reset your environment to connect to the Symphony cluster instead of the DE cluster. Continue to connect from the DE installation environment.

You can only use this command when the current installation environment is set to DE.

Your environment uses the DE libraries, and work submitted from client applications or commands issued runs on the Symphony cluster in subsequent command sessions.

soamswitch symde

Use this command to reset your environment to connect to the DE cluster instead of the Symphony cluster. Continue to connect from the DE installation environment.

You can only use this command when the current installation environment is set to DE.

Your environment uses the DE libraries, and work submitted from client applications or commands issued runs on the DE cluster in subsequent command sessions.

soamswitch sym *Symphony_dir* -v *version*

Use this command to reset your environment to connect to the Symphony cluster from the Symphony installation environment, regardless of your current environment.

You can only use this command if Symphony is installed on this machine.

Your environment uses the Symphony libraries, and work submitted from client applications or commands issued runs on the Symphony cluster in subsequent command sessions.

Symphony_dir

Specify the path to the directory in which Symphony is installed.

Note:

You cannot specify a UNC path. For example, do not specify \\hosta\share\.

-v *version*

Specifies the version of Symphony to switch to.

Use this option when multiple versions of Symphony are installed on the same machine.

Reset from DE environment and cluster to Symphony grid environment and cluster

soamswitch sym C:\EGO -v 4.1

soamswitch sym *SymphonyDE_dir* -v *version*

Use this command to reset your environment to connect to the Symphony cluster from the DE installation environment, regardless of your current environment.

You can only use this command if DE is installed on this machine.

Your environment uses the DE libraries, and work submitted from client applications or commands issued runs on the Symphony cluster in subsequent command sessions.

SymphonyDE_dir

Specify the path to the directory in which DE is installed.

Note:

You cannot specify a UNC path. For example, do not specify \\hosta\share\.

-v *version*

Specifies the version of Symphony DE to switch to.

Use this option when multiple versions of Symphony DE are installed on the same machine.

Reset from Symphony environment to the DE environment while connecting to the Symphony cluster

```
soamswitch sym C:\SymphonyDE\DE4.1
```

soamswitch symde *SymphonyDE_dir* -v *version*

Use this command to reset your environment to connect from the DE installation environment to the DE cluster, regardless of your current environment.

Your environment uses the DE libraries, and work submitted from client applications or commands issued runs on the DE cluster in subsequent command sessions.

SymphonyDE_dir

Specify the path to the directory in which Symphony DE is installed.

Note:

You cannot specify a UNC path. For example, do not specify \\hosta\share\.

-v *version*

Specifies the version of Symphony DE to switch to.

Use this option when multiple versions of Symphony DE are installed on the same machine.

Reset from Symphony grid environment and cluster to the DE environment and cluster

```
soamswitch symde C:\SymphonyDE\DE4.1
```

soamswitch info

The output of the command indicates your current environment and whether you are connecting to DE cluster or a Symphony cluster.

The output of the command indicates whether your installation environment is set to Symphony or DE.

View my current environment

View whether the environment is currently set to connect to a Symphony cluster.

```
soamswitch info
```

-h

Prints command usage to stdout and exits.

-V

Prints product version to stdout and exits.

Related files

The following configuration files specify which cluster to connect to. The paths shown result from using the default installation directories.

- `C:\SymphonyDE\DE41\conf\ego.conf`—specifies which Symphony cluster to connect to from a DE installation environment
- `C:\SymphonyDE\DE41\conf\vem_resource.conf`—specifies which DE cluster to connect to from a DE installation environment
- `C:\EGO\kernel\conf\ego.conf`—specifies which Symphony cluster to connect to from a Symphony installation environment

soamunreg

Unregisters an application and deletes the application profile, preventing more sessions from being started for this application.

Synopsis

soamunreg *application_name* [**-f**] [**-s**] [**-u** *user_name*] [**-x** *password*]

soamunreg -h | **-V**

Description

Use `soamunreg` to unregister an application.

Unregistering an application does the following:

- Terminates existing sessions and tasks (running and suspended)
- Disables the application
- Releases all resources allocated to the application
- Removes the application profile from the system

After unregistering the application, remove the application files in `$EGO_CONFDIR/../../profiles` on Linux, `%EGO_CONFDIR%\..\..\profiles` on Windows.

You can optionally save historic data associated with the application

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Options

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

You can view the application name using `soamview app`.

-f

Forces the action, bypassing prompts to unregister an enabled application with running workload and to save the historical data. This option assumes you want to unregister the application immediately.

-s

Saves the application historical data.

-u user _name

Specifies the name of the user to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Unregister an application

Unregister an application, `sampleApp`:

```
soamunreg sampleApp
```

soamview

Displays information about applications, sessions, and tasks, and displays the application profile.

Synopsis

soamview *subcommand* [*options*]

soamview *subcommand* **-h**

soamview -h | **-V**

Description

Use `soamview` to view information about applications, sessions, and tasks, and to view the application profile.

Symphony DE does not verify the optional user name and password. Symphony DE users can control all sessions and applications in their environment.

In Symphony, the information `soamview` displays depends on your user ID authorization, as follows:

- The cluster administrator can view all information.
- The consumer administrator can view information for all applications associated with that consumer. To specify an absolute consumer name that contains spaces, enclose the name in double quotes (" ").
- Users can view information for only those applications their user privileges allow, the applications they submitted.

To view large amounts of information (when output exceeds one screen), pipe or redirect the command output to a file.

The application totals displayed by `soamview` reflect the values of counters that start when the application is enabled or when a new session manager starts for this application (in a failover event). These counters are reset when the application is disabled or the session manager exits. The session and task totals `soamview` displays reflect counters in session and task runtime and historical data.

Note:

The information listed for SI startup failures lists the last five hosts on which a service instance failed to start. If more than five hosts experienced SI startup failures, only the last five hosts are listed.

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Subcommand synopsis

app *application_name* [**-l** | **-p**] [**-u** *user_name*] [**-x** *password*]

app [**-c** *consumer_ID*] [**-s** "**all**|**disabled**|**enabled**"] [**-u** *user_name*] [**-x** *password*]

session *application_name* [**-s "all**|**open**|**closed**|**suspended**|**aborted**"] [**-t** *session_tag*] [**-U** *submission_user*] [**-n** *counter*] [**-u** *user_name*] [**-x** *password*]

session *application_name***:***session_ID* [**-l**] [**-u** *user_name*] [-x password]

task *application_name***:***session_ID* [**-s "***task_state***"**] [**-t** *task_tag*] [**-n** *counter*] [**-u** *user_name*] [**-x** *password*]

task *application_name*:*session_ID*:*task_ID* [**-l**] [**-u** *user_name*] [**-x** *password*]

app *application_name* [-l | -p] [-u *user_name*] [-x *password*]

Displays brief and summary information for a specific application.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-l

Provides detailed information.

-p

Displays the application profile in XML format.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only, the password specified here overrides the password entered in `soamlogon`.

View information for an application

Display summary and brief information for the `sampleApp` application.

soamview app sampleApp

app [-c *consumer_ID*] [-s "all | disabled | enabled"] [-u *user_name*] [-x *password*]

Displays information about all enabled and disabled applications the user manages.

In Symphony DE, users have access to information on all applications and do not need to supply the application name, consumer ID, user name, or password.

In Platform Symphony, users without cluster or consumer privileges must provide an application name. Users can view information only for applications, sessions, and tasks they submitted.

-c consumer_ID

Specifies the consumer ID. The consumer ID is the same as it appears in the application profile. Enclose the consumer ID in double quotes (" ") if it contains spaces. The consumer ID starts with a forward slash.

-s all

Displays information for all applications in all states.

-s enabled

Displays information for all enabled applications.

-s disabled

Displays information for all disabled applications.

-s "enabled | disabled"

Displays information for all enabled and disabled applications. Enclose the states in double quotes (" ") and use | as a separator. For example, -s "enabled|disabled".

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Display information for all applications in the consumer

Display summary and brief information for all applications in the /SampleApplications/SOATesting consumer.

soamview app -c /SampleApplications/SOATesting -s all

session *application_name* [-s "all | open | closed | suspended | aborted"] [-t *session_tag*] [-U *submission_user*] [-n *counter*] [-u *user_name*] [-x *password*]

Displays information for sessions in various states. When used without the -s option, displays only information for open sessions.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-s all

Specifies that information for sessions in all states is required.

-s session_state

Specifies that information for all sessions in one or more of the specified states is required. To specify multiple states, enclose the states in double quotes (" ") and use | as a separator. For example, -s "open|suspended".

Only the first three letters of the state need to be specified.

You can specify an abbreviated form of the session state:

- aborted—abrt
- open—open
- suspended—susp
- closed—clsd

Tip:

The `soamview session -s aborted` command displays information only for sessions that were aborted during the time period specified by lastingPeriod in the application profile.

-t session_tag

Specifies the string that is used for identification purposes to query the session.

-U submission_user

Displays information for sessions submitted by the user specified.

-n counter

Returns the maximum number of records specified.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Display session information for sessions submitted by a specific user

Display information for 40 open or suspended sessions for the sampleApp application that were submitted by userabc.

soamview session sampleApp -s "open|susp" -U userabc -n 40

Display session information for sessions that match the session tag

Display information for open or closed sessions for the sampleApp application that match the "myTag" session tag.

soamview session sampleApp -s "open|closed" -t myTag

session *application_name*:*session_ID* [-l] [-u *user_name*] [-x *password*]

Displays brief and summary information for a specific session of an application.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

session_ID

Specifies the Platform Symphony-assigned ID of the session as reported by `soamview session`.

-l

Provides detailed information.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Display information for a specific session of an application

Display detailed information for a specific session of the sampleApp application.

soamview session sampleApp:101 -l

task *application_name*: *session_ID* [-s "*task_state*"] [-t *task_tag*] [-n *counter*] [-u *user_name*] [-x *password*]

Displays brief and summary task information for a specific task of an application.

application_name

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

session_ID

Specifies the Platform Symphony-assigned ID of the session as reported by `soamview session`.

-s task_state

Displays information for all tasks in the state specified.

'You can specify the full task state name or the first three letters of the task state:

- pending
- running
- done
- error
- canceled

-t task_tag

Displays information for all tasks that match the task tag.

The task tag supports the use of wildcard characters to represent specific string patterns, as follows:

Note:

Wildcard characters must be surrounded by quotes.

- "*" represents 0 or more characters

  For example, to display all tasks in the `done` state (including tasks with and without tags), enter:

  soamview task symping:901 -t "*" -s done
- "?" represents 1 character

  For example, to display only tasks in the `done` state that have tags, enter:

  soamview task symping:901 -t "?*" -s done

The task tag also supports the use of reserved character "!" to indicate no task tag. For example, to display all tasks in the `done` state that do not have tags, enter:

soamview task symping:901 -t "!" -s done

Note:

The "!" tag is a standalone reserved character and cannot be used to negate a regular expression.

-n counter

Returns the maximum number of records specified.

-u user_name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Display information for specific tasks

Display summary and brief task information for running tasks tagged as myTask.

soamview task sampleApp:101 -s running -t myTask

task *application_name*: *session_ID*:*task_ID* [-l] [-u *user_name*] [-x *password*]

Displays brief and summary information for a specific task.

application

Specifies the name of the application. The application name is the same as it appears in the application profile. Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

session_ID

Specifies the Platform Symphony-assigned ID of the session as reported by `soamview session`.

task_ID

Specifies the Platform Symphony-assigned ID of the task as reported by `soamview task`.

-l

Provides detailed information.

-u user _name

Specifies the name of the user to connect to Platform Symphony for this command. If you are already logged on to Platform Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

Display information for a specific task

Display summary and brief information for task ID 4899.

soamview task sampleApp:101:4899

symexec

Symexec supports the running of remote execution tasks using the Symphony infrastructure. An execution task is a child process executed by a Symphony service instance using a command line specified by a Symphony client.

Synopsis

symexec create [**-a** *application_name*] [**-e** *env1=value1*[,*env2=value2*,...]] [**-p** *preexecution_command*] [**-q** *postexecution_command*] [**-u** *user_name*] [**-x** *password*]

symexec send -s *session_ID* [**-a** *application_name*] [**-e** *env1=value1*[,*env2 =value2*,...] [**-p** *preexecution_command*] [**-q** *postexecution_command*] [**-u** *user_name*] [**-x** *password*] *command*

symexec fetch -s *session_ID* **-n** *counter* [**-a** *application_name*] [**-t** *timeout_in_seconds_for_fetch*] [**-u** *user_name*] [**-x** *password*]

symexec close -s *session_ID* [**-a** *application_name*] [**-u** *user_name*] [**-x** *password*]

symexec run [**-a** *application_name*] [**-e** *env1=value1*[,*env2=value2*,...]] [**-p** *preexecution_command*] [**-q** *postexecution_command*] [**-t** *timeout_in_seconds_for_fetch*] [**-u** *user_name*] [**-x** *password*] *command*

symexec -h | **-V**

Description

Use `symexec` as a Symphony client application to run executables, such as scripts, applications, or system calls. You cannot use symexec to run interactive graphical user interface programs. `symexec` requires an application and a service. The default, out-of-box application is `symexec4.1`.

- Use `symexec create` to create an execution session in which to run execution tasks. The session stays open until it is explicitly closed.
- Use `symexec send` to send an execution task command in the execution session.
- Use `symexec fetch` to retrieve the statuses of finished execution tasks in the execution session. The statuses are retrieved as return codes or exception description and error codes.
- Use `symexec close` to close the execution session.
- Use `symexec run` to create an execution session, run a command, retrieve the results and close the session.

To run an execution conmand on Windows, specify the full name including the file extension.

Note:

It is recommended that you do not terminate the client (for example, using Ctrl + C). This could result in loss of tasks. If interrupts are required, specify a short timeout for fetch value.

Important:

If a symexec fetch command is in progress while a fetch or send command is issued for the same session from another command prompt window, the original fetch operation will abort and the session will detach from the original client

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Options

-a *application=name*

Specifies the name of the application to use. If you do not specify an application name, the application defaults to `symexec4.1`. If you specify an application name, it must match the application name in the corresponding application profile, and the application must be registered.

-e *env=value*

Sets the environment variables to be passed to the execution session. 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 comma.

-p *preexecution_command*

Specify a pre-execution command to run before running the executable command. If the pre-execution command is successful, the executable command is run. The command you specify must exist in the same location on every compute host on which it may run, or be accessible from the compute host. Specify an absolute path, or set the path in the PATH environment variable.

-q *postexecution_command*

Specify a post-execution command to run when the executable command completes successfully. The command you specify must exist in the same location on every compute host on which it may run, or be accessible from the compute host. Specify an absolute path, or set the path in the PATH environment variable.

-u *user_name*

Specify the user ID under which to run the execution session.

-x *password*

Specify the password to authenticate the user ID.

-s *session_ID*

Specifies the unique ID of the session to operate upon.

-n *counter*

Specifies the number of tasks for which to fetch results.

If there are less than *n* task results available to be retrieved, `symexec fetch` waits until it can fetch all *n* tasks, or until the value set for `timeout_in_seconds_for_fetch` is reached.

If another `symexec send` or `symexec fetch` command is issued for the same session ID, the existing symexec fetch client is terminated, because only one client connection can be made to the session at a time.

Note:

Any task can only be fetched just once.

-t *timeout_in_seconds_for_fetch*

Specifies the number of seconds to wait for the session results.

If you do not specify a timeout, `symexec` waits indefinitely until all requested task outputs are received. If you specify 0 seconds, `symexec` returns immediately with or without the task output.

If the timeout expires and all of the results have still not been received, one of the following happens, depending on the subcommand the timeout is specified for:

- `symexec fetch`

  Keep the session open and continue to run the task.
- `symexec run`

  Close the session and terminate the task.

command

Specifies the execution command to run. The executable you specify must exist in the same location on every compute host on which it may run, or be accessible from the compute host. Specify an absolute path, or set the path in the PATH environment variable.

Restriction:

You cannot run a Windows internal command or a UNIX shell built-in command unless it is contained in a script.

Create an execution session to run a single command

```
symexec run -u user01 -x 12345 mycmd.exe
```

Creates an execution session using the default application `symexec4.1`, runs `mycmd.exe,` retrieves the results, and closes the session.

Create an execution session to run multiple commands

```
symexec create -u user01 -x 12345
symexec send -s 123 -u user01 -x 12345 mycmd.exe
symexec fetch -s 123 -n 1 -u user01 -x 12345
symexec send -s 123 -u user01 -x 12345 mycmd1.exe
```

- Creates an execution session using the default application `symexec4.1`, and returns a session ID
- Sends mycmd.exe to the session
- Retrieve the results of the command
- Sends another command, mycmd1.exe to the session. The session remains open to receive other commands.

Retrieving stdout and stderr from execution tasks

The results from an execution task can be in the form of an exit code with optional stdout and stderr data files from the execution task's command. To retrieve the stdout and stderr data

from a completed execution task, you must change its command to run under a shell so the command can redirect the stdout and stderr data to file(s). The file(s) can then be retrieved by either copying the file from the remote host (for example, with an FTP command) or by using a shared file system location.

The followingexamples demonstrate how to retrieve the stdout and stderr data from Windows and Linux hosts.

Windows:

In this example, the command will take the output of the "`dir` c:\" command and place it in the `dir_content.txt` file. The command can be entered in the Start > Run textbox. The **cmd /c** opens a command shell and closes it after processing the string.

cmd /c dir c:\ > c:\temp\dir_content.txt

Linux:

In this example, the command will take the output of the `"ls /"` command and place it in the `root_content.txt` file. The **sh -c** opens a command shell, which reads the commands from the string.

sh -c 'ls / > /tmp/root_content.txt'

symping

Sends workload to a cluster to test and verify that Symphony components are working and responsive.

Synopsis

symping

[**-f** *configuration_file*]

[**-u** *user_name*]

[**-x** *password*]

[**-a** *application_name*]

[**-s** *session_type*]

[**-l** *[summary_file*]]

[**-T** *taskoverhead*]

[**-i** *input_msg_size_in_bytes*]

[**-o** *output_msg_size*_in_bytes]

[**-r** *task_processing_time*_in_milliseconds]

[**-c** *common_data_size_in_bytes*]

[**-m** *number_of_tasks_per_testrun*]

[**-n** *number_of_testruns*]

[**-t** *client_type*]

[**-g** *session_tag*]

[**-k** *task_tag*]

[**-d**] [**-z**] [**-p**][**-j**]

symping -h | **-V**

Description

Use the `symping` Symphony client to send test workload Symphony to check that the cluster is working. You can also use this client to send different types of workload.

Without any options, `symping` tests for a complete workload cycle. In this type of test, workload is run through the cluster with full recoverability enabled (session and history). This type of run provides minimum, maximum, and average round-trip times. The average processing time averaged across all hosts is also displayed.

The default complete workload cycle configuration is the following:

- number of sessions—1
- number of tasks—20
- session type—RecoverableAllHistoricalData
- task processing time—50 milliseconds
- input message size—64 bytes
- output message size—128 bytes

The `symping` Symphony client is part of the Symphony diagnostic application. Application components are installed with the Symphony DE, and with Symphony. The diagnostic application is registered on all compute hosts in the `/SymTesting/Symping41` consumer. The diagnostic application `symping` is composed of the following:

- A service—sympingservice. The service sleeps for the task processing time. The service is always preloaded and holds one slot in the ComputeHosts group, even when idle. Use the -r and -o options to change the task processing time or the output message size.
- An application profile—`symping4.1.xml`.
- Location of `symping` executables and configuration:
  - Windows
    - 32-bit: `%SOAM_HOME%\4.1\win32-vc7\bin`
    - 64-bit: `%SOAM_HOME%\4.1\w2k3_x64-vc7-psdk\bin`
  - Linux

    Located in the architecture-specific directory.
    - 32-bit: For example, `$SOAM_HOME/4.1/linux2.4-glibc2.2-x86/bin`
    - 64-bit: For example, `$SOAM_HOME/4.1/linux2.6-glibc2.3-x86_64 /bin`

-h

Prints command usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

symping options

-f configuration_file

Path and configuration file to use. The configuration file must be accessible from the machine on which the symping client is running.

Use this option to specify options in a configuration file instead of specifying them on the command line. If options are specified on the command line and a configuration file is specified, command-line options override those in the configuration file.

You can find a sample configuration file `symping.conf` in the bin directory in which the symping client is located.

-u user _name

Not applicable to Symphony DE.

Specifies the name of the user to connect to Symphony for this command. The user must have access to the consumer to which the application is registered.

If you are already logged on to Symphony using `soamlogon`, for this command only the user name specified here overrides the user name entered in `soamlogon`.

-x password

Not applicable to Symphony DE.

Specifies the user password to connect to Symphony for this command. If you are already logged on to Symphony using `soamlogon`, for this command only the password specified here overrides the password entered in `soamlogon`.

-a application_name

The name of the application that symping submits the workload to. The application name is the same as it appears in the application profile.

To run symping, the symping service binary must be configured in the application profile (it is configured and deployed out of box). If you have a mixed cluster, refer to your installation guide to ensure you have made the necessary modifications for symping to run.

Important:

The application name should have "symping" in its name. If you copy and rename this application, keep "symping" in the file name so that you can distinguish the applications related to the Symping client from those that are not.

Enclose the application name in double quotes (" ") if it contains spaces. Precede the application name with two dashes (--) if it starts with a dash.

-s session_type

Specifies the type of the session to create. The session type is coded in the symping client and is indicated in the application profile. Including history and recoverability increases the task and session overhead but more likely simulates a production environment.

Possible session types are:

- RecoverableAllHistoricalData: Session is recoverable and task and session history are set to "All". This is the default session type when you run symping without any options.
- RecoverableNoHistoricalData: Session is recoverable but no history (session or task) is kept.
- UnrecoverableAllHistoricalData: Session is unrecoverable and no history (session or task) is kept.
- UnrecoverableNoHistoricalData: Session is unrecoverable and no host history (task or session) is kept. This is the default session type if you choose to run symping configured for task overhead with the -T taskoverhead option.

-l [summary_file]

If you specify -l without a summary file name, creates a summary file for symping output in the directory in which symping is run with the name `symping.summary.`*`date_stamp.hhmmss`*.

If you specify the path and name to assign to the summary file, generates output in the specified summary file. If the summary file exists, appends output to the existing summary file.

-T taskoverhead

Optimizes the symping configuration to calculate task overhead—the Symphony overhead for processing a single task.

The default configuration is the following:

- number of sessions—1
- number of tasks—20
- session type—UnrecoverableNoHistoricalData
- task processing time—0 milliseconds
- input message size—1024 bytes
- output message size—1024 bytes
- common data—0 bytes

Results include overhead for longest and shortest task, average overhead, and average processing time.

-i input_msg_size_in_bytes

Specifies the size of input messages for all tasks in the session, in bytes.

If you do not specify this option, the input message size for tasks is 64 bytes.

-o output_msg_size_in_bytes

Specifies the size of all output messages for all tasks in the session, in bytes.

If you do not specify this option, the output message size for tasks is 128 bytes.

-r task_processing_time_in_milliseconds

Specifies how long each task in the session runs, in milliseconds.

Task processing time is a time duration. Task processing time is the pure task processing time, without overhead. It is the time spent on the onInvoke() call. The clock is started when SIM calls onInvoke() in the service code. The clock ends when the onInvoke() returns.If you do not specify the task processing time, tasks run for 50 milliseconds.

-c common_data_size_in_bytes

Specifies the size of the common data that is associated with the session and passed from the client to the service, in bytes.

If you do not specify the size of common data, there is no common data passed from the client application to the service.

-m number_of_tasks_per_testrun

Specifies the number of tasks that are sent per test run.

If you do not specify the number of tasks, the client application creates one session with 20 tasks.

-n number_of_testruns

Specifies the number of test runs that are created. A test run indicates sending input and retrieving output.

For example, in one test run with 10 tasks, 1 session is created and 10 tasks sent.

In 2 test runs with 10 tasks, 1 session is created, 10 tasks are sent and output retrieved, then 10 more tasks are sent and output retrieved.

If you do not specify the number of test runs, the client application creates 1 test run.

-t client_type

Specifies whether tasks are sent and output retrieved synchronously or asynchronously. Valid values are sync and async.

If you do not specify the client type, tasks are sent and output retrieved asynchronously.

-g session_tag

Adds the specified session tag to any session created by symping. A session tag that is shared among sessions provides the ability to query or control these related sessions with a single action.

-k task_tag

Adds the specified task tag to all tasks submitted with symping. The task tag can later be used to filter only those tasks that share the given tag.

-d

Does not display detailed output for each task.

If you do not use this option, detailed output for each task is displayed.

-z

Specifies to consume actual CPU cycles on compute hosts.

If you do not specify this option, the service sleeps for the specified task processing time, not consuming actual CPUs on the compute host.

-p

Symphony DE only.

Displays compute host status in Symphony DE.

-j

Generates a service log file on compute hosts on which the symping service runs. You can retrieve and view the log file with the Platform Management Console.

The log is written to:

- Windows— `%SOAM_HOME%\work\`*application_name*`\`*session_ID*
- Linux— `$SOAM_HOME/work/`*application_name*`/`*session_ID*

The log file is named according to session and task IDs: *sessionID.taskID*.

Without this option, no service log is generated.

Output

The following are definitions for terms used in the symping output:

- Task overhead:

  Duration. Symphony overhead for processing a single task. It is calculated as follows:Task roundtrip minus task processing
- Task processing:

Duration.Pure task processing time (no overhead): time spent on the service onInvoke() call. The clock is started when SIM calls onInvoke() in the service code. The clock ends when onInvoke() returns.

- Task roundtrip:

  Duration. Time for a full task cycle (overhead + processing) of a single task. It is the time from the send task input call to the return of the task output.

  Task round trip is defined as:Task output received minus task input sent
- Total task roundtrip:

  Duration. Time for a full task cycle (overhead + processing) for all tasks in a session. It is the time from the send task input call to the return of the task output.

  Total task roundtrip is defined as:Last task output received minus first task input sent
- Test roundtrip:

  Duration. Clock is started when the user runs the symping command. Clock ends when symping returns with the client application's results and returns control to user.This time duration includes initialization, connection time, all phases before session creation time.

Examples

Test for a complete workload cycle

```
symping
```

Test for task overhead

Use the -T taskoverhead option to optimize configuration to calculate system overhead.

```
symping -T taskoverhead
```

Create a summary file with symping output

Use the -l option to generate a summary file with symping output.

```
symping -l
```

Generate a service log file on compute hosts

Use the -j option to generate a service log file on compute hosts.

```
symping -j
```

Run one session with 10 tasks

```
symping -m 10
```

Run 2 sessions, 1000 tasks each, 1 millisecond in length

To run more than one session, you need to run `symping` twice

```
symping -m 1000 -r 1
```

```
symping -m 1000 -r 1
```

Run 30 tasks, keep the session open, then run 30 more tasks

```
symping -m 30 -n 2
```

Use common data of 150 bytes, input messages of 20 bytes, output messages of 20 bytes

```
symping -c 150 -i 20 -o 20
```

Run one session with 10 tasks, 1 millisecond in length, with common data of 150 bytes, and input and output messages of 20 bytes

```
symping -m 10 -r 1 -c 150 -i 20 -o 20
```

PART II

Application Profile

Application profile reference

About application profiles

The application profile is used to change application parameters without requiring code changes. The application profile defines application behavior within Symphony. It provides the information Symphony needs to run services and manage workload. Each application name in the profile must be unique in the cluster and associated with one consumer. At any time, there can be only one enabled application per consumer.

You can find example application profiles in the samples directories. You can also refer to the application profile templates to create your own application profiles. You can find templates in the `$SOAM_HOME/4.1/samples/Templates` directory.

Application profile fragment

With dynamic application updates, you can add a new service by registering a *profile fragment* (part of an application profile) that contains either new session types in the SessionTypes section, new Service sections, or both. When you register a profile fragment that adds a session type or Service section, running workload continues to run. The session director updates the session manager with the new version of the application profile.

Note:

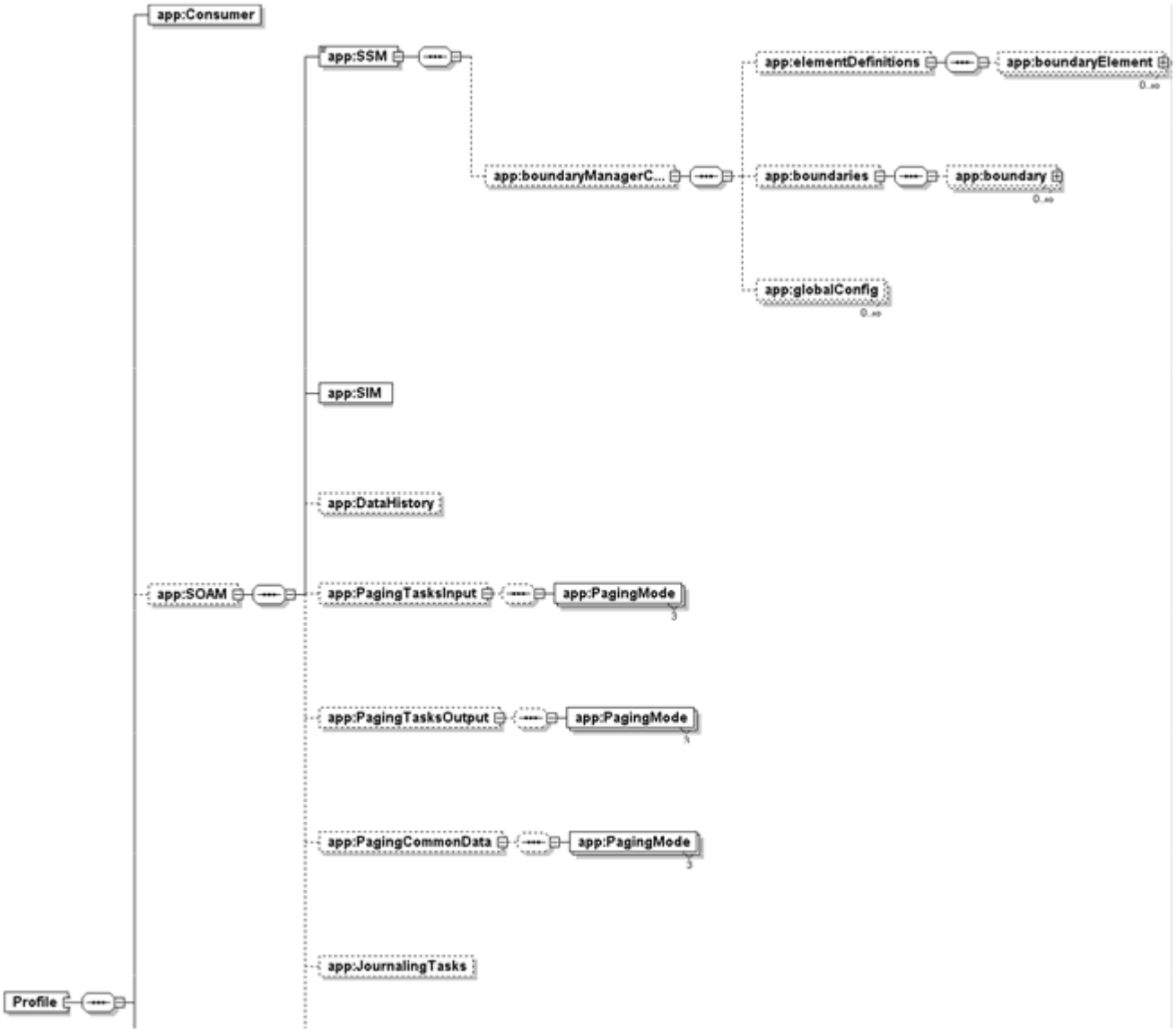
You can add, update, or remove session types or Service sections or both by modifying an existing application in PMC using **Dynamic Configuration Update**. In this case, you are not required to create fragments manually. See the PMC Help or Platform Application Development Guide, available from the Platform Knowledge Center, for more details.

You can refer to the profile fragment templates to create your own profiles. You can find templates in the `$SOAM_HOME/4.1/samples/Templates` directory. However, this is not a recommended way for updating application parameters.

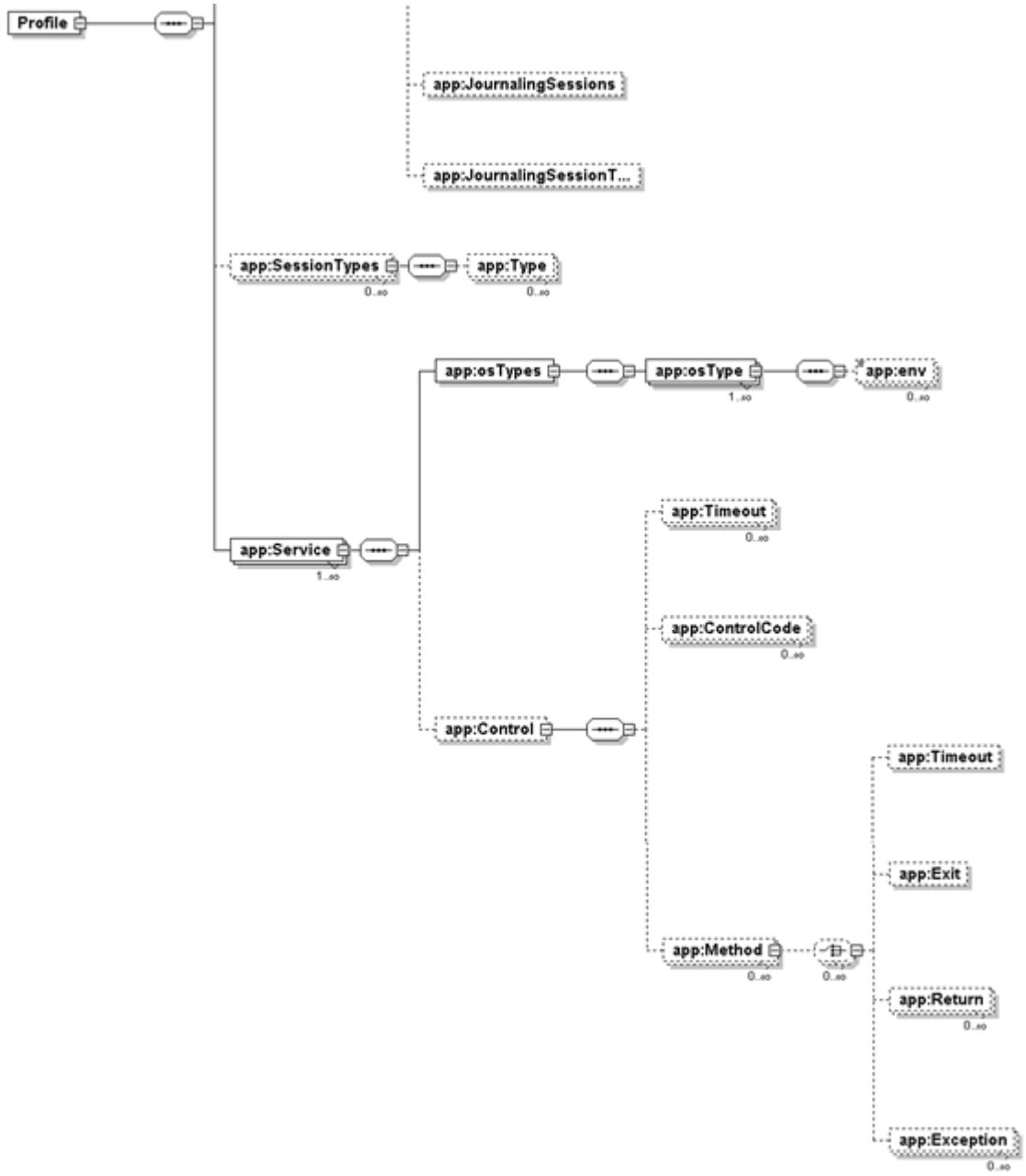
When you update a Service section or session type, the session manager only terminates workload that uses the updated service(s) and/or session type(s).

Application Profile Structure

Application profile structure (part 1)



Application profile structure (part 2)



Profile

name

For internal system use only. Not used by the system.

version

Specifies the version of the application profile.

Where used

Profile

Required/Optional

Optional

Valid values

3.1 or higher

Default value

3.1

Example

version="4.1"

Related Attributes

Version (SOAM section)—Symphony middleware version.

Consumer section

Settings related to the application and resources assigned to the consumer associated with the application.

applicationName

Name of the application associated with the consumer. The application name must be unique within the cluster.

Enclose the application name in double quotes if it contains spaces, for example, name="Commodity Derivatives App".

Where used

Consumer

Required/Optional

Required

Valid values

In the Platform Management Console: Accepts "a-z, A-Z, 0-9 or spaces. Do not use "`all`" as an application name—it is a reserved word.

On the command line: Accepts all ASCII printable characters except "\ / : * ? ' | & < or >". Does not accept leading or trailing spaces in application names. Do not use "`all`" as an application name—it is a reserved word.

consumerId

Name and path of the consumer with which to associate this application. The consumer must already exist in Symphony. The consumer you associate with the application determines how Symphony allocates resources to the application.

The consumer ID includes the path to the consumer within the consumer tree.

The consumer must be a leaf consumer. A leaf consumer always resides at the lowest level of the consumer tree.

A unique consumer ID must be specified in Symphony, but it is not required to be pre-defined in SymphonyDE. SymphonyDE uses the consumer ID to associate applications and service packages. Only one application can be enabled within a consumer ID and service packages must be deployed to the same consumer ID.

Where used

Consumer

Required/Optional

Required

Valid values

Alphanumeric string that does not include any special characters.

The consumer ID begins with a forward slash (/) and each level is separated from the next with a forward slash.

Example

consumerId="/New York Cluster/Capital Markets/Derivatives"

resourceGroupName

Resource group from which resources are requested to run service instance managers and services to perform computations for clients. The resource group should contain compute hosts, not management hosts. The resource group must be an existing resource group in EGO.

Not required for Symphony DE.

Where used

Consumer

Required/Optional

Optional

Default value

" " (empty)—all resource groups which are associated with the consumer will be used.

Related attributes

- numOfPreloadedServices—Defines the number of service instances prestarted for an application cannot be greater than the number of resources available in the resource group, defined by resourceGroupName
- resReq—Defines the resources an application requires

resReq

String that describes criteria for selecting resources allocated to a consumer for compute hosts.

Ignored in Symphony DE.

Where used

Consumer

Required/Optional

Optional

Default value

select (!mg)

Related attributes

- resourceGroupName—Defines the resources available to an application

Synopsis

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

Description

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

Note:

If you specify a resource requirement, ensure you indicate "`select(!mg)`" in addition to your resource requirement string. This is so that the system only selects compute hosts to run work, not management hosts.

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

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

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.

resource_name operator value

resource_name

The following resources can be used as selection criteria.

Static resources

Static resources are built-in resources that represent host information that does not change over time, such as the maximum RAM available to user processes or the number of processors in a machine. Most static resources are determined at start-up time, or when hardware configuration changes are detected.

Static resources can be used to select appropriate hosts based on binary architecture, relative CPU speed, and system configuration.

Note:

The resources ncpus, ncores, nprocs, nthreads, maxmem, maxswp, and maxtmp are not static on UNIX hosts that support dynamic hardware reconfiguration.

Index	Measures	Units	Determined by
type	host type	string	configuration
model	host model	string	configuration
hname	host name	string	configuration
cpuf	CPU factor	relative	configuration
server	host can run remote jobs	Boolean	configuration
rexpri	execution priority	nice(2) argument	configuration
ncpus	number of processors	processors	LIM
ndiks	number of local disks	disks	LIM
maxmem	maximum RAM	MB	LIM
maxswp	maximum swap space	MB	LIM
maxtmp	maximum space in `/tmp`	MB	LIM

CPU factor (cpuf)

The CPU factor is 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. CPU factors are defined by the cluster administrator. For multiprocessor hosts, the CPU factor is the speed of a single processor.

Server

The server static resource is Boolean. It has the following values:

- 1 if the host is configured to run jobs from other hosts.
- 0 if the host is a client for submitting jobs to other hosts.

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
>	>	a > b	1 if a is greater than b, 0 otherwise
<	<	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 &gt; 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 &gt; 500 &amp;&amp; maxswp &gt;=300)
```

order

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

order synopsis

order(*order_string***)**

order description

The order string acts on the results of a select 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 the result of an arithmetical expression.

The result orders the resources from smallest to largest.

order_string

Specify an arithmetic expression that expresses the desired outcome.

You can create an expression using operators and numbers.

Use the following operators:

+

add

–

subtract-

*****

multiply

/

divide

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)
```

policy

Specifies the workload scheduling policy used. The scheduling policy determines how and when Symphony allocates resources to run services for the application.

Where used

Consumer

Required/Optional

Optional

Default value

R_Proportion

Valid values

- R_Proportion

Summary	A new session can get CPU slots immediately without waiting for current sessions to finish. Sessions simultaneously share a portion of the total CPU slots assigned to the application. CPU slots are proportionally assigned to sessions based on priority.
Disadvantage	This scheduling policy can have performance issues if it takes a long time to bind a service to a session since the service instances may toggle among open sessions.
Initial slot allocation	Symphony assigns CPU slots first to higher priority sessions to ensure they finish faster. CPU slots are assigned proportionally to sessions according to their priority. CPU factors are taken into account during allocation. The most powerful idle CPU is allocated first to higher priority sessions. The CPU factor of the compute host is defined in the `conf\vem_resource.conf` file under the Symphony DE installation directory, and is detected by Platform EGO in Symphony cluster when Symphony starts on the host.
Slot reallocation based on workload	Yes. When a session does not need all slots allocated to it, the slots are released and reallocated to other sessions. Slots are only reallocated to sessions with pending tasks.
Higher priority sessions can take slots away	Yes. As soon as a higher priority session needs more resources, Symphony can allocate slots that are being used by lower priority sessions to the higher priority session after the tasks running on the slots are finished.
Service instance unbound from session	Service instances are unbound from the session based on workload, when there are not enough pending tasks for all the service instances currently allocated.

- R_MinimumServices

Summary	Use this scheduling policy when you are using common data so that service instances will be reused for tasks in the same session, eliminating the need to reload data for each task. With this scheduling policy, a minimum number of CPU slots is allocated to a session regardless of workload or priority.

Initial slot allocation	Symphony attempts to allocate slots to the session to satisfy the minimum configured number of service instances. If there are enough slots to meet the minimum number of instances for all open sessions: • Minimum instances are allocated to sessions based on session submission time. • Leftover slots are allocated to sessions proportionally based on session priority and submission time. If there are not enough slots to meet the minimum number for all sessions: • Higher priority sessions get all minimum instances satisfied first. CPU factors are taken into account. CPU slots are evenly distributed among sessions based on speed so that all sessions get a proportion of the fastest CPUs.
Slot reallocation based on workload	Partial. A minimum number of CPU slots is always allocated to a session, regardless of workload. Additional CPU slots are distributed proportionally to other open sessions with pending tasks according to priority and submission time. When there is no workload in the session, CPU slots not serving the minimum required service instances are distributed proportionally to other open sessions with pending tasks according to priority and submission time. Sessions with no pending tasks do not receive any slots, except the required minimum service instances.
Higher priority sessions can take slots away	Partial. A session always retains the required CPU slots to serve the minimum number of service instances, regardless of session priority. CPU slots serving the minimum number of service instances are not released. CPU slots not serving the minimum number of service instances are released and distributed to other sessions proportionally based on priority.
Service instance unbound from session	The minimum number of service instances are always bound to the session until the session is suspended, killed, or closed. Service instances additional to the minimum number are unbound from the session as needed, based on the priority and pending workload of all sessions.

Related attributes

SessionType:

- priority—Determines how are tasks are assigned resources.
- minServices—Determines the minimum number of CPU slots required for sessions.

Consumer:

- taskHighWaterMark—Defines the maximum ratio of unprocessed tasks to CPU slots before more resources are requested.
- taskLowWaterMark—Defines the minimum ratio of unprocessed tasks to CPU slots before unused resources are released.

- resourceBalanceInterval—Defines when Symphony next checks to determine whether resources need to be requested or released from Platform EGO.
- sessionSchedulingInterval—Determines when resources are balanced among sessions.

taskHighWaterMark

Applies to the whole application. Ratio is used to determine whether more CPU slots need to be requested to meet the service level agreement of the application. The ratio is the total number of pending and running (unprocessed) tasks in open sessions to CPU slots allocated.

If an application has a taskHighWaterMark of 5, it means that for every 5 unprocessed tasks, 1 CPU slot is requested.

In Symphony DE, the number of CPU slots on a host is configured in the `conf/vem_resource.conf` file under the Symphony DE installation directory.

In Symphony, the number of CPU slots on a host is configured in the `ResourceGroups.xml` file and can be configured from PMC.

Where used

Consumer

Required/Optional

Optional

Valid values

integer, 0.00000 to 9999999999.99999

Default value

1.0

Related attributes

- resourceBalanceInterval—Defines when Symphony next checks to determine whether resources need to be requested or released from Platform EGO
- taskLowWaterMark—Defines the minimum ratio of unprocessed tasks to CPU slots before unused resources are released

taskLowWaterMark

Applies to the whole application. Ratio is used to determine whether idle CPU slots are released and made available for other applications to use. The ratio is the total number of pending and running (unprocessed) tasks in open sessions to CPU slots allocated.

This enables the application to be configured to avoid charges for unnecessary or unused resources.

For example, you set taskLowWaterMark to 2.0. You have thousands of unprocessed tasks. 100 CPU slots are allocated to the Session Manager.

- When the number of unprocessed tasks reaches 400, the ratio of unprocessed tasks to CPU slots is 400/100= 4.0. Session Manager does not release any CPU slots.
- When the number of unprocessed tasks reaches 200, the ratio of unprocessed tasks to CPU slots is 200/100=2.0. Session Manager does not release any CPU slots.

- When the number of unprocessed tasks drops below 200, Session Manager releases slots until the ratio becomes 2.0 or greater.

The value of the low-water mark should be less than or equal to the high-water mark.

If taskLowWaterMark is 1.0 or less, any resources the application releases must be idle. If taskLowWaterMark is equal to 0, the application never returns resources voluntarily.

Where used

Consumer

Required/Optional

Optional

Valid values

integer, 0.00000 to 9999999999.99999

Default value

0.0

Related attributes

- resourceBalanceInterval—Defines when Symphony next checks to determine whether resources need to be requested or released form Platform EGO
- taskHighWaterMark—Defines the maximum ratio of unprocessed tasks to CPU slots before more resources are requested

resourceBalanceInterval

Minimum number of seconds between checks to determine whether the session manager should release unused resources or request additional resources for the application.

Where used

Consumer

Valid values

1 or more

Required/Optional

Optional

Default value

5 seconds

Related attributes

- resReq—Defines the resources an application requires
- resourceGroupName—Defines the resource groups available to an application
- sessionSchedulingInterval—Determines when resources are balanced among sessions

- taskHighWaterMark—Defines the maximum ratio of unprocessed tasks to CPU slots before more resources are requested
- taskLowWaterMark—Defines the minimum ratio of unprocessed tasks to CPU slots before unused resources are released

sessionSchedulingInterval

Minimum number of milliseconds between checks to determine whether compute resources need to be reallocated among sessions.

The system reallocates resources according to application session priority, scheduling policy, and pending tasks.

Where used

Consumer

Required/Optional

Optional

Valid values

1 or more

Default value

500 milliseconds

Related attributes

- policy—Defines how resources are shared.
- priority—Determines how are tasks are assigned resources.
- taskLowWaterMark—Defines the minimum ratio of unprocessed tasks to CPU slots before unused resources are released
- taskHighWaterMark—Defines the maximum ratio of unprocessed tasks to CPU slots before more resources are requested

preStartApplication

Specifies whether the session manager, service instance manager, and service instances are prestarted for enabled applications when Symphony is started, or when an application is enabled.

Prestarting the session manager provides quicker response to the first client request for this application. Prestarting the service instances manager and service instances provides quicker response to tasks.

Where used

Consumer

Required/Optional

Optional

Valid values

true | false

Default value

false

Related attributes

- numOfPreloadedServices—Defines the number of service instances prestarted for an application cannot be greater than the number of resources available in the resource group, defined by resourceGroupName

numOfPreloadedServices

Used with pre-start application. Defines the number of service instances that are prestarted if preStartApplication is true.

One service instance manager is started for every service instance. One service instance can run on each CPU slot.

Where used

Consumer

Required/Optional

Optional

Valid values

1 or greater

Default value

1

Related attributes

- preStartApplication—Services are preloaded only if preStartApplication is true
- resourceGroupName—Defines the resources available to an application

transientDisconnectionTimeout

Number of seconds the session manager waits for the client to reconnect when the connection between the client and session manager is broken.

When abortSessionIfClientDisconnect is true, sessions which are open on the client connection are aborted if a disconnected client does not reconnect to the session manager.

The transientDisconnectionTimeout counter is reset when a client connects or re-connects to the session manager. Reconnection is done by the Symphony Client API, which is transparent to the client application.

Where used

Consumer

Required/Optional

Optional

Valid values

1 or more

Default value

10 seconds

Related attributes

- abortSessionIfClientDisconnect—Specifies whether the session is aborted if the session manager detects that the connection between the client and the session manager is broken.

flushDataAsap

Used for recoverable sessions. Specifies whether or not the session manager caches data before writing to disk.

When set to true, data is not cached, it is immediately written to disk. When set to false, data is cached before it is written to disk. Note that the setting in flushDataAsap does not affect any operating system configuration. You need to disable write-behind caching for your operating system. On Windows, see http://support.microsoft.com/kb/247485 for more details. On Linux, contact your system administrator.

Important:

Setting this parameter to true may substantially degrade performance because system cannot continue with its next operation until the data is written to the disk.

Where used

Consumer

Required/Optional

Optional

Valid values

true | false

Default value

false

ioRetryDelay

Specifies the amount of time to wait, in seconds, before retrying an I/O operation after a previous failure.

Where used

Consumer

Required/Optional

Optional

Valid values

Non-negative integer

Default value

1 second

ssmRecoveryTime

For internal system use only.

tttvhjbhjhj

SOAM section

The SOAM section contains configuration related to system behavior such as session manager, service instance manager, and how Symphony writes data to disk.

version

The version of the Symphony middleware.

Where used

SOAM

Required/Optional

Required—When version in the Profile section is configured to 3.2 or higher.

Optional—When version in the Profile section is configured to 3.0 or 3.1.

Valid values

3.1 or higher

Example

version="4.1"

SOAM>SSM section

Configuration related to session manager behavior.

startUpTimeout

Number of seconds to wait for the session manager to start up before the session director considers it as timed out.

When the process times out, the session director requests EGO to replace the session manager host and tries again to start a new session manager.

The session manager is started when Symphony is started if prestartApplication is true for the enabled application. Otherwise, the session manager is started when a client connects to an enabled application, which does not have running session manager and prestartApplication is false.

Where used

SOAM > SSM

Required/Optional

Optional

Valid values

1 or more

Default value

60 seconds

Related attributes

preStartApplication—Influences when the session manager and service instance manager are started, when Symphony starts or when Symphony receives a request from a client of this application

shutDownTimeout

Number of seconds to wait for the session manager to shut down before the session director considers it as timed out and restarts the session manager.

The session manager should shut down within the timeout period when the related application is unregistered or disabled.

Where used

SOAM > SSM

Required/Optional

Optional

Valid values

1 or more

Default value

300 seconds

disableFlowControl

For internal system use only.

resReq

String that describes criteria for selecting resources allocated to the session director to start the session manager.

This is not applicable for Symphony DE.

Where used

`SOAM > SSM`

Required/Optional

Optional

Default value

" " (empty)—SSM can run on any hosts in the resource groups configured in `resourceGroupName` in the SOAM > SSM section. By default, the `resourceGroupName` is `ManagementHosts`.

Synopsis

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

Related Attributes

- resourceGroupName in SOAM > SSM—Resource group from which resources are requested to run session managers.

Description

A resource requirement string describes the criteria for defining a set of resources to run session managers. Session managers should run on management hosts. If you have changed the default value of resourceGroupName, you must indicate the select string `"select(mg)"` so that only management hosts are selected to run the session managers. Running the session manager on a management host is essential for reliability and recovery.

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

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

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.

resource_name operator value

resource_name

The following resources can be used as selection criteria.

Static resources

Static resources are built-in resources that represent host information that does not change over time, such as the maximum RAM available to user processes or the number of processors in a machine. Most static resources are determined at start-up time, or when hardware configuration changes are detected.

Static resources can be used to select appropriate hosts based on binary architecture, relative CPU speed, and system configuration.

Note:

The resources ncpus, ncores, nprocs, nthreads, maxmem, maxswp, and maxtmp are not static on UNIX hosts that support dynamic hardware reconfiguration.

Index	Measures	Units	Determined by
type	host type	string	configuration
model	host model	string	configuration
hname	host name	string	configuration
cpuf	CPU factor	relative	configuration
server	host can run remote jobs	Boolean	configuration
rexpri	execution priority	nice(2) argument	configuration
ncpus	number of processors	processors	LIM
ndiks	number of local disks	disks	LIM
maxmem	maximum RAM	MB	LIM
maxswp	maximum swap space	MB	LIM
maxtmp	maximum space in `/tmp`	MB	LIM

CPU factor (cpuf)

The CPU factor is 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. CPU factors are defined by the cluster administrator. For multiprocessor hosts, the CPU factor is the speed of a single processor.

Server

The server static resource is Boolean. It has the following values:

- 1 if the host is configured to run jobs from other hosts.
- 0 if the host is a client for submitting jobs to other hosts.

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
/	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.

Example: Simple selection expression using static resources

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

```
select(maxmem &gt; 500)
```

Example: 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 &gt; 500 &amp;&amp; maxswp &gt;=300)
```

order

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

order synopsis

order(*order_string***)**

order description

The order string acts on the results of a select 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 the result of an arithmetical expression.

The result orders the resources from smallest to largest.

order_string

Specify an arithmetic expression that expresses the desired outcome.

You can create an expression using operators and numbers.

Use the following operators:

+

add

–

subtract-

*****

multiply

/

divide

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)
```

resourceGroupName

Resource group from which resources are requested to run session managers. The resource group should be a management host resource group and must be an existing resource group in EGO.

Not required for Symphony DE.

Where used

- `SOAM >SSM`

Required/Optional

Optional

Default value

If the resource group is not defined, the system selects resources from the ManagementHosts group to start the session manager.

Related attributes

- resReq in SOAM > SSM—Specifies the criteria for selecting resources to start the session manager. Resources will only be selected from resource groups specified by resourceGroupName .

workDir

Absolute path in which the session manager process stores data for operations, including paging, journaling and history files.

Paging, journaling, and history data files are saved in subdirectories under the workDir if a relative path is configured for the paging, journaling, and data history paths.

This is not applicable in Symphony DE. In Symphony DE the working directory of the session manager is:

- Windows— `%SOAM_HOME%\work`
- Linux— `$SOAM_HOME/work`

Where used

- SOAM > SSM

Required/Optional

Optional

Valid values

Any accessible absolute path on the network or local disk. To facilitate the session manager recovery, workDir should be set to a directory on a shared file system.

Default value

By default, the session manager working directory is the same for all applications. If applications are configured to write to the same directory, the operating system user accounts assigned to start the session manager must have access to that directory. It is recommended that all operating system execution users who start the session managers be in the same primary user group.

If a shared directory is configured in the cluster, the default working directory for the session manager points to the shared location in the host on which session manager runs:

- `${EGO_SHARED_TOP}/soam/work`

If there is no shared directory configured in the cluster, then the default working directory points to the local location in the host on which session manager runs:

- `Windows–%EGO_CONFDIR%\..\..\soam\work`
- `Linux–$EGO_CONFDIR/../../soam/work`

SOAM>SSM>boundaryManagerConfig>elementDefinitions>boundaryElement

For internal system use only.

name

For internal system use only.

description

For internal system use only.

minValue

For internal system use only.

maxValue

For internal system use only.

SOAM>SSM>boundaryManagerConfig>elementDefinitions>boundaryElement>additionalConfig

For internal system use only.

name

For internal system use only.

value

For internal system use only.

SOAM >SSM > boundaryManagerConfig>boundaries>boundary

elementName

Specifies how the system responds to changes in the amount of virtual memory or virtual address space on the session manager host.

- AvailableMemory—The boundary manager monitors the amount of virtual memory available on the host. Virtual memory is physical memory plus swap memory that is available on the host.
- AvailableVirtualAddressSpace—The boundary manager monitors the amount of virtual address space the session manager can use to map memory. It is possible for the host to have more than enough memory available, which is physical memory plus swap memory, but the process cannot access anymore because it has no available address space to map this memory to.

Where used

SOAM > SSM > boundaryManagerConfig > boundaries > boundary

Required/Optional

Optional

Valid values

AvailableMemory | AvailableVirtualAddressSpace

Related attributes

- name (event)—Multiple boundary events are associated with each elementName
- value (event)—A value is assigned to each boundary event

SOAM >SSM > boundaryManagerConfig>boundaries>boundary>event

name (event)

Specifies the name of the boundary event that triggers a system response when the value of the event is reached.

The system monitors the boundaries defined, available memory, and available virtual address space, to ensure the robust functionality of the session manager. Messages are logged to the session manager log file and administrators are notified through event notification when new event are triggered..

Memory low conditions can be caused by:

- Running too many memory -consuming processes on the session manager host
- Too many unprocessed input and output messages
- Too many unfinished sessions or tasks

The following describes system reactions when a boundary limit is reached:

Boundary event	Messages	Workload processing changes	System processing changes
BEV_PROACTIVE	INFO	Delayed response for incoming requests	Starts scheduling common data and , and input and output messages for paging.
BEV_SEVERE	WARNING	• Delayed response for incoming client requests • The session manager allows clients to create new sessions and submit new tasks. These actions were prohibited by the session manager after the BEV_CRITICAL event was triggered.	Assigns higher priority to paging write requests, memory is released once the messages are paged out
BEV_CRITICAL	WARNING	• New connections from the client to the session manager are rejected • Delayed response for requests already in process • New session creations and tasks submissions are held until a lower level event is triggered such as BEV_SEVERE or BEV_PROACTIVE	Assigns higher priority to paging write requests

Boundary event	Messages	Workload processing changes	System processing changes
BEV_HALT	FATAL	• All new session and client requests are rejected • Processing of existing requests is stopped	• Runs in maintenance mode • Only cluster administrator can access workload through the console • Once memory drops down, and BEV_SEVERE or BEV_PROACTVE is triggered, the session manager resumes work

Memory low conditions can be corrected by:

- Running the session manager on a dedicated management host
- Adding more physical memory or increasing the amount of assigned swap memory
- Terminating any faulty sessions that are consuming excessive amounts of memory
- Terminating any nominal sessions to release memory

Where used

SOAM > SSM > boundaryManagerConfig > boundaries > boundary > event

Required/Optional

Optional

Valid values

BEV_PROACTIVE | BEV_SEVERE | BEV_CRITICAL | BEV_HALT

Related attributes

- elementName—The name of the system resource monitored by the boundary manager
- value (event)—The minimum value the system resource can reach before the event is triggered

value (event)

Specifies the value of a boundary event as a percentage of the entire resource available.

When the availability of the resource monitored falls below the boundary event value, it triggers a system response to ensure the session manager has enough memory to function.

Where used

SOAM > SSM > boundaryManagerConfig > boundaries > boundary > event

Required/optional

Optional

Valid values

0 or more

Default values

The following describes the element names, event names, and their default boundary values:

elementName / event name	BEV_PROACTIVE	BEV_SEVERE	BEV_CRITICAL	BEV_HALT
AvailableMemory	50	40	25	15
AvailableVirtualAddressSpace	50	40	25	15

Related attributes

- elementName—Specifies the resource to which the boundary applies.
- name (event)—Specifies the boundary event for which a value is assigned.

SOAM>SSM>boundaryManagerConfig>globalConfig

For internal system use only.

name

For internal system use only.

value

For internal system use only.

SOAM>SIM

blockHostOnTimeout

Used with the startUpTimeout (SOAM > SIM) attribute. Defines whether to block this host for the application when the service instance manager times out on process startup. If set to true, when the service instance manager times out on startup, the host is added to the blocked host list and is no longer used for the application unless the host is explicitly unblocked through the Platform Management Console.

Symphony DE blocks slots rather than hosts. Slots can be unblocked by disabling and enabling your application.

Where used

SOAM > SIM

Required/Optional

Optional

Valid values

true | false

Default value

true, host is added to the blocked host list for the application upon timeout

Related attributes

startUpTimeout (SOAM > SIM)

blockHostOnVersionMismatch

For internal system use only.

startUpTimeout

Number of seconds to wait for the service instance manager to start up before the session manager considers it as timed out. This parameter works in conjunction with blockHostOnTimeout.

After a session manager starts a service instance manager, if the service instance manager cannot contact the session manager within the `startUpTimeout,` the session manager requests a new host from EGO and tries to start a new service instance manager on the new host.

Where used

SOAM > SIM

Required/Optional

Optional

Valid values

1 or more

Default value

60 seconds

Related attributes

- blockHostOnTimeout—Determines whether the host is blocked if timeout occurs on SIM startup.

SOAM>DataHistory

Parameters related to sessions and tasks historical data storage.

pollFrequency

Specifies the frequency, in seconds, at which the history files are polled to determine if the current file should be archived.

Where used

SOAM > DataHistory

Required/Optional

Optional

Valid values

1 to 10 (inclusively)

Default value

10 seconds

Related attributes

- fileSwitchSize—Defines, according to file size, when old history files are archived and new history files are started
- fileSwitchTime—Defines, according to file age, when old history files are archived and new history files are started

fileSwitchSize

Maximum size, in megabytes, the history file can reach before the file is archived at the next polling interval.

When the history file reaches the maximum size specified, the file is renamed by appending a time stamp to the file name.

Data history file names have the following format:

*applicationName*_session.soamdb or *applicationName*_task.soamdb

After renaming the file names would be similar to the following:

*applicationName*_session.soamdb.*yyyy_mm_dd_hh_mm_ss_x*

*applicationName*_task.soamdb.*yyyy_mm_dd_hh_mm_ss_x*

Where used

SOAM > DataHistory

Required/Optional

Optional

Valid values

1 to 100 (inclusively)

Default value

100 MB

Related attributes

pollFrequency—The size of the file is checked at the time interval defined by pollFrequency

fileSwitchTime

Maximum number of hours that can elapse before the history file is archived at the next polling interval.

When the history file reaches the maximum age specified, the file is renamed by appending a time stamp to the file name.

Data history file names have the following format:

*applicationName*_session.soamdb or *applicationName*_task.soamdb

After renaming the file names would be similar to the following:

*applicationName*_session.soamdb.*yyyy_mm_dd_hh_mm_ss_x*

*applicationName*_task.soamdb.*yyyy_mm_dd_hh_mm_ss_x*

Where used

SOAM > DataHistory

Required/Optional

Optional

Valid values

1 to 168(1 week)

Default value

24 hours

Related attributes

pollFrequency—The age of the file is checked at the time interval defined by pollFrequency

lastingPeriod

Number of hours the archived history files are retained before deleted.

Where used

SOAM > DataHistory

Required or optional

Optional

Range or valid values

1 to 168 (1 week)

Default value

96 hours (4 days)

lastingPeriodInSeconds

For internal system use only.

path

Absolute or relative path in which session and task historical files are stored. If relative path is specified, the directory is located under session manager working directory. The path should be accessible by both session director and session manager, so that the session and task historical data can be retrived by command line and PMC.

Note:

Ensure the user account running the session director (cluster administrator), and the user account running the session

manager for the application (os execution account assigned to the consumer) has access to the directory.

By default, history files are saved under the following locations:

- For Symphony,
  - Windows—`%EGO_CONFDIR%\..\..\soam\work\history`
  - Linux—`$EGO_CONFDIR/../../soam/work/history`
- For Symphony DE,
  - Windows—`%SOAM_HOME%\work\history`
  - Linux—`$SOAM_HOME/work/history`

If you specify an absolute path, all the history files are saved in the specified path. An absolute path must be accessible by both session director and the session manager for this application.

Where used

- SOAM > DataHistory

Required/Optional

Optional

Valid values

Any absolute or relative path on network or local disk which can be accessed by both session director and session manager.

Default value

By default, data history files are saved in the "history" directory under the session manager working directory.

Related attributes

- workDir (SOAM > SSM)—the working directory of session manager.

delimiter

For internal system use only.

SOAM>PagingTasksInput

Parameters related to task input message paging configurations.

path

Directory where paging files for task input are stored. The path to the file can be specified as a relative or absolute path. If a relative path is specified, paging files are saved in a subdirectory relative to the session manager working directory. The specified directory must be accessible from all management hosts to facilitate session manager failover.

Where used

- SOAM > PagingTasksInput

Required/Optional

Optional

Valid values

Any accessible absolute or relative path on network or local disk that can be accessed from all management hosts.

Default values

By default, paging files are saved in the "paging" directory under the session manager working directory.

Related attributes

- workDir (SOAM > SSM)—the working directory of session manager.

blockSize

Specifies the number of bytes per block for paging. The block size should be large enough to contain the largest message. If the block size is not large enough, the data is spread across multiple blocks, which can slow down paging processes.

Consider the size of the largest input message when determining the optimal blocksize for paging input tasks

Where used

- SOAM > PagingTasksInput

Required/Optional

Optional

Valid values

1 or more

Default value

4096 bytes (4KB)

diskSpace

Maximum amount of disk space, in bytes, dedicated to paging task input messages.

Where used

SOAM > PagingTasksInput

Required/Optional

Optional

Valid values

1 or more

Default values

4294967296 bytes (4 GB)

bitmapBits

For internal system use only.

pmeType

For internal system use only.

SOAM>PagingTasksInput>PagingMode

For internal system use only.

name

For internal system use only.

readPriority

For internal system use only.

writePriority

For internal system use only.

removePriority

For internal system use only.

SOAM>PagingTasksInputNonRec

For internal system use only.

path

For internal system use only.

Where Used

For internal system use only.

Required/Optional

For internal system use only.

Valid values

For internal system use only.

Default values

For internal system use only.

SOAM>PagingTasksOutput

Parameters related to task output message paging configurations.

path

Directory where paging files for task output are stored. The path to the files can be specified as a relative or absolute path. If a relative path is specified, paging files are saved in a

subdirectory relative to the session manager working directory. The specified directory must be accessible from all management hosts to facilitate session manager failover.

Where used

- SOAM > PagingTasksOutput

Required/Optional

Optional

Valid values

Data files can be stored on any storage device accessible on the network. To facilitate failover, Symphony working files should be stored on a shared file system.

Default values

By default, paging files are saved in the "paging" directory under the session manager working directory.

Related attributes

- workDir (SOAM > SSM)—the working directory of session manager.

blockSize

Specifies the number of bytes per block for paging. The block size should be large enough to contain the largest message. If the block size is not large enough, the data is spread across multiple blocks, which can slow down paging processes.

Consider the size of the largest output message when determining the optimal blocksize for paging output tasks

Where used

- SOAM > PagingTasksOutput

Required/Optional

Optional

Valid values

1 or more

Default value

4096 bytes (4KB)

bitmapBits

For internal system use only.

diskSpace

Maximum amount of disk space, in bytes, dedicated to paging task output messages.

Where used

SOAM > PagingTasksOutput

Required/Optional

Optional

Valid values

1 or more

Default value

4294967296 bytes (4 GB)

pmeType

For internal system use only.

SOAM>PagingTasksOutput>PagingMode

For internal system use only.

name

For internal system use only.

readPriority

For internal system use only.

writePriority

For internal system use only.

removePriority

For internal system use only.

SOAM>PagingTasksOutputNonRec

For internal system use only.

path

For internal system use only.

Where Used

For internal system use only.

Required/Optional

For internal system use only.

Valid values

For internal system use only.

Default values

For internal system use only.

SOAM>PagingCommonData

Parameters related to common data paging configurations.

path

Directory where the files are stored for common data. The path to the files can be specified as a relative or absolute path. If a relative path is specified, common data files are saved in a subdirectory relative to the session manager working directory. The specified directory must be accessible from all management hosts to facilitate session manager failover.

Where used

- SOAM > PagingCommonData

Required/Optional

Optional

Valid values

Data files can be stored on any storage device accessible on the network. To facilitate failover, Symphony working files should be stored on a shared file system.

Default values

By default, paging files are saved in the "paging" directory under the session manager working directory.

Related attributes

- workDir (SOAM > SSM)—the working directory of session manager.

blockSize

Specifies the number of bytes per block for paging. The block size should be large enough to contain the largest message. If the block size is not large enough, the data is spread across multiple blocks, which can slow down paging processes.

Consider the size of the largest common data message when determining the optimal block size for paging common data.

Where used

- `SOAM > PagingCommonData`

Required/Optional

Optional

Valid values

1 or more

Default value

102400 bytes (100KB)

diskSpace

Specifies the maximum amount of disk space, in bytes, dedicated to paging.

Where used

SOAM > PagingCommonData

Required/Optional

Optional

Valid values

1 or more

Default values

8589934592 bytes (8GB)

bitmapBits

For internal system use only.

pmeType

For internal system use only.

SOAM>PagingCommonData>PagingMode

For internal system use only.

name

For internal system use only.

readPriority

For internal system use only.

writePriority

For internal system use only.

removePriority

For internal system use only.

SOAM>PagingCommonDataNonRec

For internal system use only.

path

For internal system use only.

Where Used

For internal system use only.

Required/Optional

For internal system use only.

Valid values

For internal system use only.

Default values

For internal system use only.

SOAM>PagingCommonDataUpdates

Parameters related to common data updates paging configurations.

path

Directory where the files are stored for common data updates. The path to the files can be specified as a relative or absolute path. If a relative path is specified, common data update files are saved in a subdirectory relative to the session manager working directory. The specified directory must be accessible from all management hosts to facilitate session manager failover.

Where used

- SOAM > PagingCommonDataUpdates

Required/Optional

Optional

Valid values

Data files can be stored on any storage device accessible on the network. To facilitate failover, Symphony working files should be stored on a shared file system.

Default values

By default, paging files are saved in the "paging" directory under the session manager working directory.

Related attributes

- workDir (SOAM > SSM)—the working directory of session manager.

blockSize

Specifies the number of bytes per block for paging. The block size should be large enough to contain the largest message. If the block size is not large enough, the data is spread across multiple blocks, which can slow down paging processes.

Consider the size of the largest common data update message when determining the optimal block size for paging common data updates.

Where used

- `SOAM > PagingCommonDataUpdates`

Required/Optional

Optional

Valid values

1 or more

Default value

102400 bytes (100KB)

diskSpace

Specifies the maximum amount of disk space, in bytes, dedicated to paging.

Where used

SOAM > PagingCommonDataUpdates

Required/Optional

Optional

Valid values

1 or more

Default values

8589934592 bytes (8GB)

bitmapBits

For internal system use only.

pmeType

For internal system use only.

SOAM>PagingCommonDataUpdates>PagingMode

For internal system use only.

name

For internal system use only.

readPriority

For internal system use only.

writePriority

For internal system use only.

removePriority

For internal system use only.

SOAM>PagingCommonDataUpdateNonRec

For internal system use only.

path

For internal system use only.

Where Used

For internal system use only.

Required/Optional

For internal system use only.

Valid values

For internal system use only.

Default values

For internal system use only.

SOAM>JournalingTasks

path

For internal system use only.

blockSize

For internal system use only.

bitmapBits

For internal system use only.

diskSpace

For internal system use only.

pmeType

For internal system use only.

SOAM>JournalingSessions

path

For internal system use only.

blockSize

For internal system use only.

bitmapBits

For internal system use only.

diskSpace

For internal system use only.

pmeType

For internal system use only.

SOAM>JournalingSessionTagConfig

path

For internal system use only.

blockSize

For internal system use only.

bitmapBits

For internal system use only.

diskSpace

For internal system use only.

pmeType

For internal system use only.

SessionTypes section

A session type defines a group of configuration settings for session attributes. Multiple session types can be defined in this section with different configuration settings for each session type. Symphony also defines a system default session type with the name "". You can add or update (or remove) a session type to an existing application dynamically using PMC.

A client application specifies the session type when creating a session. The session type that the client specifies must be configured in this section or it should use the system default.

Type

The Type element defines a session type, which is a group of configuration settings for session attributes.

name

Session type name is an identifier for the group of configuration settings. A client application specifies the session type name when creating a session. When a session is created, the configuration settings for the specified session type are applied to that session.

The session type is optional. If you leave this parameter blank " " or do not set a session type, system default values are used for session attributes. If you specify a session type in the client application, you must also configure the session type in the application profile—the session type name in your application profile and session type you specify in the client must match. If you use an incorrect session type in the client and the specified session type cannot be found in the applicatin profile, an exception is thrown to the client.

Where used

SessionTypes > Type

Required/Optional

Required

Valid values

1—256 alphanumeric characters, including all printable characters.

serviceName

Name of service that is used to execute the workload for the sessions in this session type. This is any name that you assign to the service to link the session type to the service definition. The name you enter here must match the name entered in the `<Service>` section.

Specify a service name only when multiple services are defined in the application profile and when you want to assign the session type a different service than the default service. In the client application you can use the appropriate API to override the service that is used to execute the workload for the session that you are creating. Refer to the API Reference for more details.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

1—256 alphanumeric characters, including all printable characters.

Default value

If only one service section is defined in the application profile, uses the service defined in the Service section by default. If more than one service is defined in the application profile, uses the service identified as the default service by default.

priority

Specifies the session priority, where 1 is the lowest priority and 10000 is the highest priority. If there are multiple open sessions, the session priority affects how resources are distributed across sessions.

The resource distribution is also affected by the scheduling policy of the session manager. For more details, refer to the `policy` attribute in the `Consumer` section.

Where used

`SessionTypes > Type`

Required/Optional

Optional

Valid values

1 to 10000

Default value

1

Related attributes

policy—Determines how resources are allocated to sessions according to the scheduling policy used

minServices

Minimum number of CPU slots required for each open session of this type. The minServices attribute is applicable only when the R_MinimumServices scheduling policy is used.

Symphony attempts to allocate the minimum required number of CPU slots to each session before proportionally allocating additional available CPUs based on

Service instances started on the minimum number of CPU slots remain bound to the session until the session is suspended, killed or closed.

Where used

Consumer > policy

Required/Optional

Optional

Valid values

0-10000

Default value

0

Related attributes

- policy—Determines how resources are allocated to sessions according to the scheduling policy used. The minServices attribute is applicable only when the R_MinimumServices scheduling policy is used.

recoverable

Specifies whether the session and its workload can be recovered if the session manager restarts or fails over.

If true, the data for tasks and other session information are retained to attempt to recover the session in a failover situation.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

true | false

Default value

false

Related attributes

- workDir (SOAM > SSM) —Defines where working files are stored

sessionRetryLimit

Specifies the number of times the session manager retries to bind the service to a session if the service fails in the `SessionEnter` or `SessionUpdate` API method. After the session experiences the specified number of retries, the session is aborted.

Note:

The retry count for both of these methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal

to 3. Therefore the SessionRetryCount should be set to a value that accounts for both SessionEnter and SessionUpdate failures.

You can specify what events you want Symphony to consider as bind failures by configuring attributes in the Service > Control > Method section for your service.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

0 or more

Default value

3

taskRetryLimit

Specifies the number of attempts to retry a task in the session if the task does not complete successfully. After a task has reached this number of retries and still does not complete successfully, the task is put into the Error state.

This attribute is used if the service fails during the API `Invoke` method.

If the value of taskRetryLimit is 3, the system makes a maximum of 4 attempts to run the task before the task fails.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

0 or more

Default value

3

taskGracePeriod

Number of milliseconds to allow a task to continue running after a session of the task is suspended by a user.

Number of milliseconds to allow the running service method to complete and to allow the service instance to clean up after the resource on which the service instance is running is reclaimed.

For clean up, Symphony will initiate the onSessionLeave() method and the onDestroyService () (if applicable) after the current running service method completes.

- If a task is running and the Invoke method completes during the task grace period, the result of that method is treated as it would be treated under normal circumstances.
- If a task is running and the Invoke method does not complete before the task grace period expires, the service instance on which the task is running is terminated and the task is requeued.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

0 or more

Default value

100 milliseconds

Related attributes

- suspendGracePeriod - If suspendGracePeriod is defined in the application profile, it overrides the value for taskGracePeriod for the suspend scenario.
- reclaimGracePeriod - If reclaimGracePeriod is defined in the application profile, it overrides the value for taskGracePeriod for the reclaim scenario.

reclaimGracePeriod

Number of milliseconds to allow the current running service method to complete and the service instance to clean up when the resource on which the service instance is running is reclaimed. If the service method and cleanup does not complete within the set time, then Symphony will terminate the instance. If the timeout has not expired, Symphony will initiate cleanup after the current running service method completes.

When this reclaimGracePeriod < EGO Consumer reclaim grace period, the reclaimGracePeriod takes effect.

When this reclaimGracePeriod >= EGO Consumer reclaim grace period, the EGO Consumer reclaim grace period takes effect.

If the Invoke method was running and completes during the applied reclaim grace period, the result of that method is treated as it would be treated under normal circumstances.

If the Invoke method was running and does not complete before the applied reclaim grace period expires, the service instance on which the task is running is terminated and the task is requeued.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

- 0 or more
- -1 indicates unlimited grace period, the EGO Consumer reclaim grace period is used instead

Default value

-1

suspendGracePeriod

Number of milliseconds to allow a task to continue running after the session for the task is suspended by a user.

A task is considered to be running on a service instance when that service instance is executing its Invoke method for that particular task.

- If the Invoke method completes during the suspend grace period, the result of that method is treated as it would be treated under normal circumstances.
- If the Invoke method does not complete before the suspend grace period expires, the service instance on which the task is running is terminated and the task is requeued.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

0 or more

Default value

100 milliseconds

taskCleanupPeriod

Number of milliseconds to allow a running task to finish executing when the corresponding session has been aborted by the system or a user, or a task is killed by a user. When a session is aborted, the running tasks of the session are canceled, and the service instance manager will not return any task output from the service instance to the session manager even if the task completes before the taskCleanupPeriod expires.

The service instances which have running tasks will not be released until the `taskCleanupPeriod` expires or the API `Invoke` call returns.

Where used

SessionTypes

Required /Optional

Optional

Valid values

0 or more

Default value

250 milliseconds

abortSessionIfClientDisconnect

Specifies whether the session is aborted if the session manager detects that the connection between the client and the session manager is broken.

- If true, and the transientDisconnectionTimeout expires, the session is aborted.
- If false, and the transientDisconnectionTimeout expires, the session is retained and its workload continues to run. The client can reconnect to session to resume its workload.

Note:

Note that if in a new connection a session that was previously disconnected is opened within the transientDisconnectionTimeout period after the original client exited abnormally, the session is not aborted even if abortSessionIfClientDisconnect is set to true.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

true | false

Default value

true

Related attributes

transientDisconnectionTimeout—Defines how long the session manager waits for a client to reconnect before it aborts the sessions that are open on the client's connection.

abortSessionIfTaskFail

Specifies whether the session is aborted if a task fails (enters the `Error` state).

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

true | false

Default value

false

persistSessionHistory

Specifies whether to persist session history. If any form of task history is retained, session history must also be retained.

Session history information includes, but is not limited to:

- Session creation and end times
- Final state reached by the session
- Summary of tasks for the session

Any unrecoverable sessions that are not in their final states are not recorded.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

none | all

Default value

all

Related attributes

- path(SOAM > DataHistory)—Session history is stored in the history files whose location is defined by path
- persistTaskHistory—If task history is retained, session history must also be retained

persistTaskHistory

Specifies whether to retain no task history, all task history, or only task history for error tasks. Session history must also be retained for task history to be retained.

- none—Does not persist task history
- all—Persists history for all tasks
- error—Persists history only for tasks that end in error

The task history contains details about the task such as run time, rerun, and rerun counts.

Any unrecoverable tasks that are not in their final states are not recorded.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

none | all| error

Default value

all

Related attributes

- path (SOAM > DataHistory)—Task history is stored in the task history files whose location is defined by path
- persistSessionHistory—If task history is retained, session history must also be retained

discardResultsOnDelivery

Specifies whether the system discards task output for the session when output has been retrieved by the client.

A value of true sets data to be discarded once output has been retrieved by the client.

A value of false sets data to be discarded only after a session is closed or aborted. Note that if set to false, outstanding output for all sessions are tracked until the session completes.

Where used

SessionTypes > Type

Required/Optional

Optional

Valid values

true | false

Default value

true

Service section

The Service element defines a group of configuration settings for service attributes. Multiple service elements can be defined in this section with different configuration settings for each service. Defining multiple services allows you to use more than one service within a single application. You can add or update (or remove) a service to an existing application dynamically using PMC.

name

Name of the service to run for this application. The service name must be unique within the application.

A client application can use the appropriate API to override the service that is used to execute workload. Refer to the API Reference for more details. The service name that the client specifies must be configured in the application profile.

Where used

Service

Required/Optional

Required

Valid values

1—256 alphanumeric characters, including all printable characters.

Related attributes

- serviceName—Name of service that is used to execute the workload for the sessions in this session type.

description

String that describes the service.

Where used

Service

Required/Optional

Optional

Valid values

1—256 alphanumeric characters, including all printable characters.

default

Identifies a service as the default service, which means that the default service is immediately started when the service instance manager starts.

Where used

Service

Required/Optional

Optional when one service section is defined.

Required when multiple service sections are defined.

Valid values

true | false

Default value

True if only one service is defined in the application profile. If only one service is defined, that service is identified as the default and is started when the service instance manager starts.

When multiple services are defined, at least one of the services default value should be set to true. Other services' default value are optional.

packageName

Name assigned to the service package during deployment. The value you specify here must match the value specified during deployment.

Where used

Service

Required/Optional

Required if you use the `SOAM_DEPLOY_DIR` variable in the Service section; optional if you do not use that variable in the Service section.

Default

None

maxOtherInstances

Applies only when multiple services are used in a single application. Defines the maximum number of other service instances that a single service instance manger can run concurrently, in addition to the current service.

When this parameter is set to 0, no other service instances started by the service instance manager can run at the same time when this service instance manager starts this service. As a result, any other running service instances managed by this service instance manager are stopped when this service is started.

When this parameter is set to a number larger than 0, the service instance manager can keep `maxOtherInstances` service instances running along with this service concurrently. Setting a value larger than 0 saves service instance loading time, because service instances are not shut down and restarted.

When a service instance manager needs to shutdown a service to satisfy maxOtherInstances, it first tries to gracefully shutdown the service instance. If the service instance does not exit when the `DestroyService` timeout (`Service > Control > Method > Timeout`, duration attribute) expires, the process is killed.

Where used

Service

Required/Optional

Optional

Default

0, only one service instance can run at a time.

Example

For example, if you have three services with different values for maxOtherInstances:

Name	maxOtherInstances	Default
S1	1	true
S2	1	false
S3	0	false

When the service instance manager starts, it starts service S1 by default. The service instance manager has one child process (S1). When service S2 is started, the service instance manager has 2 child processes (S1 and S2). If service S3 is started, the service instance manager has only one child process, which is S3 (S1 and S2 are shutdown by the service instance manager).

When service S1 is started again, the service instance manager has only one child process, which is S1 (S3 is shutdown by the service instance manager because it's maxOtherInstances is 0).

deploymentTimeout

Maximum time, in seconds, to wait before soamdeploy checks the connection with the repository server.

Where used

Service

Required/Optional

Optional

Default

300 seconds

Valid values

0—2147483647

deploymentRetryCount

Specifies the number of times soamdeploy retries to connect to the repository server and download the service package from the repository server. After the service experiences the specified number of retries, the service is aborted.

Where used

Service

Required/Optional

Optional

Default

3

Valid values

0 or Positive integer.

debugSetting

Helps you to use a debugger to step through the service code and find any errors in the service logic or in the environment.

For the customized debugSetting, you can configure the specific events that you want Symphony to detect by specifying the customizedDebugAction for each of these events.

Where used

Service

Required/Optional

Optional

Default

none

Valid values

none | customized | full

none—never generates debug logs

customized—generates debug logs upon detection of specific customizable events

full—always generate debug logs

cleanupTimeout

Maximum time, in seconds, to allow a service instance to cleanup if an application is disabled or unregistered, ora middleware component becomes unavailable.

Cleanup allows the running service method to gracefully finish executing. If the service method and cleanup does not complete within the set time, then Symphony will terminate the instance.

Where used

Service

Required/Optional

Optional

Default

60 seconds

Valid values

Positive integer

Service >osTypes >osType

Configuration related to the operating system in which services run.

name

Operating system environment in which to run the service. Symphony uses this information to determine where the executable for the service is located and set up the environment variables for the service instance.

Symphony supports:

- all—This configuration is applied if the osType for the allocated resource is not configured in the application profile.
- LINUX86—A 32-bit Linux-based environment, such as RedHat Linux
- NTX86—A 32-bit Windows-based environment, such as Windows 2000, Windows NT, and Windows XP
- X86_64—A 64-bit Linux-based environment
- NTX64—A 64-bit Windows-based environment
- SOL64—A Solaris SPARC based environment
- SOLX8664—A Solaris x86-64 based environment
- User-defined operating-system types, defined in the `ego.shared` file

Note:

While specifying PATH, ensure that you use the proper path delimiter (colon or semi-colon) according to the osType.

Where used

- Service > osTypes > osType

Required/Optional

Required

Valid values

all | LINUX86 | X86_64 | NTX86 | NTX64 | SOL64 | SOLX8664

logDirectory

Used for log retrieval through the Platform Management Console. Path to the directory in which logs are written by services for this application. The path can be any desired path and must be the same on all compute hosts.

Configure logDirectory, fileNamePattern, and subDirectoryPattern for each operating system type defined in your service section.

Note:

Log retrieval is only available in Symphony on the grid. Log retrieval is not available in Symphony DE.

Where used

Service > OsTypes > osType

Required/Optional

Optional

Default value

- Linux—$SOAM_HOME/work
- Windows—%SOAM_HOME%\work

Related attributes

- fileNamePattern—File naming convention for service log files written to the logDirectory.
- subDirectoryPattern—Convention for naming subdirectories in the logDirectory.

fileNamePattern

Used for log retrieval through the Platform Management Console. File naming convention for service log files written to the logDirectory. Specify how your files are named. When retrieving logs for a specific task, the system looks for log files that partially match the file name pattern. If you are going to name files with the task ID or session ID, specify the variables %taskId% or %sessionId%.

Configure logDirectory, fileNamePattern, and subDirectoryPattern for each operating system type defined in your service section.

Note:

Log retrieval is only available in Symphony on the grid. Log retrieval is not available in Symphony DE.

Where used

Service > OsTypes > osType

Required/Optional

Optional

Valid values

alphanumeric string

Special variables: %taskId%, %sessionId%

Default value

empty, log retrieval is not enabled

Example

If log files have, for example, the service name, then sessionID, and task ID, for example, your log file name is sampleService_123_1.log, you can specify:

sampleService_%sessionId%_%taskId%.log

The system matches files containing the session ID and task ID.

Related attributes

- logDirectory—Path to the directory in which logs are written by services for this application.
- subDirectoryPattern—Convention for naming subdirectories in the logDirectory.

subDirectoryPattern

Used for log retrieval through the Platform Management Console. Convention for naming subdirectories in the logDirectory. If you want to name subdirectories according to session ID or task ID use the variables %sessionId%, %taskId%. When retrieving logs, the system looks for log files in subdirectories that partially match the sub-directory pattern.

Configure logDirectory, fileNamePattern, and subDirectoryPattern for each operating system type defined in your service section.

Note:

Log retrieval is only available in Symphony on the grid. Log retrieval is not available in Symphony DE.

Where used

Service > OsTypes > osType

Required/Optional

Optional

Valid values

alphanumeric string

Special variables: %sessionId%, %taskId%

Default value

empty, no subdirectories exist

Example

If you want to name subdirectories according to session ID, specify, without any backslashes:

%sessionId%

The system looks for logs in subdirectories named according to the session ID.

Related attributes

- logDirectory—Path to the directory in which logs are written by services for this application.
- fileNamePattern—File naming convention for service log files written to the logDirectory.

startCmd

Path to the program executable for the service. If you deploy a service package using Symphony deployment, the directory where your service package is extracted can be referred to as `${SOAM_DEPLOY_DIR}`. If you need to reference a file in your service package, use the `${SOAM_DEPLOY_DIR}` variable.

For example, on Windows, if in your service package you have the directory structure `\myservice\myservice.exe`, indicate `${SOAM_DEPLOY_DIR}\myservice\myservice.exe` in startCmd.

For example, on Linux, if in your service package you have the directory structure `/myservice/myservice`, indicate `${SOAM_DEPLOY_DIR}/myservice/myservice`. in startCmd.

Note:

To run a Windows .bat script, you need to specify a special syntax to run the batch file in command shell. For example:

```
<osType name=NTX86" startCmd="cmd /c cmd /c
install.bat"/>
```

Where used

Service > osTypes > osType

Required/Optional

Required

Valid values

Any location accessible on your network or local disk.

workDir

Working directory of the service instance.

Where used

- Service> osTypes > osType

Required/Optional

Optional

Valid values

Any location on local disk.

Default value

The default working directory for the service is:

- ${SOAM_HOME}/work

preExecCmd

When using a deployment tool other than Symphony soamdeploy, the command specified here will run on the compute host and can be used to copy the service program to that host.

Multiple commands can be run by calling a script or batch file containing the commands. The path to this command must be defined by an environment variable.

preExecCmd is called only when the `packageName` is not configured in the service section.

Where used

Service > osTypes > osType

Required/Optional

Optional

Default value

"" (empty)—No pre-execution command will be executed

Service > osTypes >osType > env

Configure envirionment variables for running service instances.

name

Name of an environment variable to set in the runtime environment of the service. You can also define environment variables so that they get substituted in the startCmd, preExecCmd, and workDir attributes.

One env statement is required for each environment variable.

No environment variables need to be set for the service.

You can refer to defined environment variables in other environment variables. For example:

<env name="ENVAR1">${SOAM_HOME}/work</env>

<env name="ENVAR2">${ENVAR1}/${VERSION_NUM }</env>

Symphony substitutes the following environment variables with system values:

- ${PATH}—Specifies the path to the relevant executable. The specified path is pre-pended in front of the system Path at runtime. For Windows, specifies the path to any dependent library, including the Symphony libraries.
- ${LD_LIBRARY_PATH}—Linux only. Specifies the path to the library where the GCC-specific Symphony files are located. The specified path is pre-pended in front of the system LD_LIBRARY_PATH at runtime
- ${SOAM_HOME}—Directory where Symphony is installed. Replaced with the value of the operating system environment variable SOAM_HOME set on the host.
- ${VERSION_NUM}—Symphony version on which the service is running.
- ${EGO_MACHINE_TYPE}—Specifies the host type installation. For example, win32-vc7 specifies a Windows machine. The variable is replaced with the value of the operating system environment variable EGO_MACHINE_TYPE set on the host.
- ${SOAM_DEPLOY_DIR}—Internal system directory in which the service package is deployed.

- ${SOAM_SERVICE_EVENT_REPLAY_LOG}— Service is driven by the events logged in the SERL file that this variable references. If this environment variable is not defined, service is driven by the SIM (as through Symphony).

Note:

EGO_MACHINE_TYPE, SOAM_DEPLOY_DIR, and SOAM_SERVICE_EVENT_REPLAY_LOG environment variables are reserved by Symphony, modification of these environment variables may result in undesirable behavior of your application.

Where used

- Service > osTypes > osType > env

Required/Optional

Required

<Service ><Control><Method> element

name

Service API method to which defined events apply.

Where used

Service > Control > Method

Required/Optional

Required

Valid values

- Register
- CreateService
- SessionEnter
- SessionUpdate
- Invoke
- SessionLeave
- DestroyService

Default value

No default value

<Service ><Control><Method><Timeout> element

Defines how long to wait for the method specified in <Method name=...> element to complete, and what action to take on service instances, sessions, or tasks, upon timeout of the method.

duration

Number of seconds to wait for the method specified in <Method name=...>to complete before a timeout is considered.

Required/Optional

Required

Valid values

Method	Valid value
Register	1 or more
CreateService	0 or more
SessionEnter	0 or more
SessionUpdate	0 or more
Invoke	0 or more
SessionLeave	0 or more
DestroyService	0 or more

Note:

The method never times out if you set the value to 0.

Default values

Method	Default value
Register	60 seconds
CreateService	0 seconds, never timeout
SessionEnter	0 seconds, never timeout
SessionUpdate	0 seconds, never timeout
Invoke	0 seconds, never timeout
SessionLeave	0 seconds, never timeout
DestroyService	15 seconds

actionOnSI

Action to take on the service instance when a timeout occurs.

Where used

Service > Control > Timeout

Required/Optional

Required

Valid values

- blockHost—When a timeout is reached on the method, terminate the running service instance on this host and do not use this host to start any other service instance for the

application. The host on which the service instance was running is added to the blocked host list for the application. This host is no longer selected to run work for the application until it is explicitly unblocked through the EGO command line or the Platform Management Console.

- restartService—When a timeout is reached on the method, terminate the service instance, start a new service instance on the same host, and recover state. There is no limit to the number of times that a service instance can be restarted.

Not all values are possible with all methods. The table below outlines possible values:

Method	Possible action
Register	• blockHost • restartService
CreateService	• blockHost • restartService
SessionEnter	• blockHost • restartService
SessionUpdate	• blockHost • restartService
Invoke	• blockHost • restartService
SessionLeave	• blockHost • restartService
DestroyService	• No action possible on the service instance

Default values

Default values differ according to method. The following table outlines default values by method.

Method	Default action
Register	• blockHost
CreateService	• blockHost
SessionEnter	• blockHost
SessionUpdate	• blockHost
Invoke	• restartService
SessionLeave	• restartService

Method	Default action
DestroyService	• No action possible

actionOnWorkload

Action to take on sessions and tasks when a timeout occurs.

Where used

Service > Control > Method > Timeout

Required/Optional

Required

Valid values

- retry—When a timeout is reached on the method, retry the method up to the number of times configured by the session and task retry limits in the application profile.

  For SessionEnter and SessionUpdate, the system attempts to bind the session to the service instance up to the sessionRetryLimit in the application profile before the session is aborted.

  Note:

  The retry count for both of these methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3. Therefore the SessionRetryCount should be set to a value that accounts for both SessionEnter and SessionUpdate failures.

  For Invoke, the system attempts to run the task up to the taskRetryLimit defined in the application profile before the task is failed.
- fail—When a timeout is reached on the method, abort the session or fail the task, and propagate errors to the client application.

  For SessionEnter or SessionUpdate, immediately abort the session. Do not retry the method.

  For Invoke, immediately fail the task. Do not retry the method.

Not all values are possible with all methods. The table below outlines possible actions for methods:

Method	Possible action
Register	• No action possible
CreateService	• No action possible
SessionEnter	• retry • fail

Method	Possible action
SessionUpdate	• retry • fail
Invoke	• retry • fail
SessionLeave	• No action possible
DestroyService	• No action possible

Default values

Default values differ according to method. The following table outlines default values by method for actions taken.

Method	Default action
Register	• No action possible
CreateService	• No action possible
SessionEnter	• retry
SessionUpdate	• retry
Invoke	• retry
SessionLeave	• No action possible
DestroyService	• No action possible

customizedDebugAction

Action to take on sessions and tasks when a timeout occurs and service debugsettion="customized".

Where used

Service > Control > Method > Timeout

Required/Optional

Optional

Valid values

- writeServiceEventReplayFiles—When Symphony detects that the specified method has timed out, it generates service event replay files to capture the relevant events that lead up to the timeout. This is the recommended setting if method timeout is an unexpected event for your service.

- none—When Symphony detects that the specified method has timed out, it does not generate service event replay files. This is the recommended setting if your service times out as a normal occurrence.

Default value

writeServiceEventReplayFiles

<Service ><Control><Method><Exit> element

Defines what action to take on service instances, sessions, or tasks when the method specified with the <Method> element exits.

actionOnSI

Action to take on the service instance when the method exits.

Where used

Service > Control > Method > Exit

Required/Optional

Required

Valid values

- blockHost—When the method exits, terminate the running service instance on this host and do not use this host to start any other service instance for the application. The host on which the service instance was running is added to the blocked host list for the application. This host is no longer selected to run work for the application until it is explicitly unblocked through the EGO command line or the Platform Management Console.
- restartService—When the method exits, terminate the service instance, start a new service instance on the same host, and recover state. There is no limit to the number of times that a service instance can be restarted.

Note:

For DestroyService, there is no action is possible on the service instance.

Default values

Default values differ according to method. The following table outlines default values by method for actions.

Method	Default action
Register	• blockHost
CreateService	• blockHost
SessionEnter	• blockHost
SessionUpdate	• blockHost

Method	Default action
Invoke	• restartService
SessionLeave	• restartService
DestroyService	• No action possible

actionOnWorkload

Action to take on sessions and tasks when the method exits.

Where used

Service > Control > Method > Exit

Required/Optional

Required

Valid values

- retry—If the service exits during execution of the specified method, retry the method up to the number of times configured by the session and task retry limits in the application profile.

  For SessionEnter and SessionUpdate, the system attempts to bind the session to the service instance up to the sessionRetryLimit in the application profile before the session is aborted.

  Note:

  The retry count for both of these methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3. Therefore the SessionRetryCount should be set to a value that accounts for both SessionEnter and SessionUpdate failures.

  For Invoke, the system attempts to run the task up to the taskRetryLimit defined in the application profile before the task is failed.
- fail—When the method exits, abort the session or fail the task, and propagate errors to the client application.

  For SessionEnter or SessionUpdate, immediately abort the session. Do not retry the method.

  For Invoke, immediately fail the task. Do not retry the method.

Not all values are possible with all methods. The table below outlines possible actions for methods:

Method	Possible action
Register	• No action possible

Method	Possible action
CreateService	• No action possible
SessionEnter	• retry • fail
SessionUpdate	• retry • fail
Invoke	• retry • fail
SessionLeave	• No action possible
DestroyService	• No action possible

Default values

Default values differ according to method. The following table outlines default values by method for actions taken.

Method	Default action
Register	• No action possible
CreateService	• No action possible
SessionEnter	• retry
SessionUpdate	• retry
Invoke	• retry
SessionLeave	• No action possible
DestroyService	• No action possible

customizedDebugAction

Action to take on sessions and tasks when the service process exits during the execution of the specified method and service debugSetting="customized"..

Where used

Service > Control > Method > Exit

Required/Optional

Optional

Valid values

- writeServiceEventReplayFiles—When Symphony detects that the service process has exited (or crashed) while executing the specified method, it generates service event replay files to capture the relevant events that lead up to the exit. This is the recommended setting if the service process exiting or crashing in the specified method is an unexpected event for your service.
- none—When Symphony detects that the service process has exited (or crashed) during execution of the specified method, it does not generate service event replay files. This is the recommended setting if your service exits as a normal occurrence.

Default value

writeServiceEventReplayFiles

<Service ><Control><Method><Return> element

Defines what action to take on service instances, sessions, or tasks when the method specified in <Method name=...> returns normally (controlCode=0), or if a control code is defined, what action to take when the method returns with the specified control code.

actionOnSI

Action to take on the service instance when the method returns normally or with the specified control code.

Where used

Service > Control > Method > Return

Required/Optional

Required

Valid values

Note:

Actions on return cannot be configured for Register or DestroyService

- blockHost—When the method returns normally or with a specified control code, terminate the running service instance on this host and do not use this host to start any other service instance for the application.

  The host on which the service instance was running is added to the blocked host list for the application. This host is no longer selected to run work for the application until it is explicitly unblocked through the EGO command line or the Platform Management Console.
- restartService—When the method returns normally or with a specified control code, terminate the service instance, start a new service instance on the same host, and recover state. There is no limit to the number of times that a service instance can be restarted.
- keepAlive—When the method returns normally or with a specified control code, take no action on the running service instance. Return an error only.

Default values

Method	Default value
Register	• No action possible
CreateService	• keepAlive
SessionEnter	• keepAlive
SessionUpdate	• keepAlive
Invoke	• keepAlive
SessionLeave	• keepAlive
DestroyService	• No action possible

controlCode

Numeric identifier that indicates to the system what action to take based on the number that is returned by the method specified in the <Method> element. There can be multiple control codes defined per method, and different actions to take based on control code.

Where used

Service > Control > Method > Return

Required/Optional

Required

Valid values

integers, positive and negative

Default value

0, default action taken on workload and service instances, as specified by default actions on actionOnSI and actionOnWorkload.

actionOnWorkload

Action to take on sessions and tasks when the method returns normally, or with a specified control code.

Where used

Service > Control > Method > Return

Required/Optional

Required

Valid values

- retry—When the method returns normally or with the specified control code, retry the method up to the number of times configured by the session and task retry limits in the application profile.

  For SessionEnter and SessionUpdate, the system attempts to bind the session to the service instance up to the sessionRetryLimit in the application profile before the session is aborted.

  Note:

  The retry count for both of these methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3. Therefore the SessionRetryCount should be set to a value that accounts for both SessionEnter and SessionUpdate failures.

  For Invoke, the system attempts to run the task up to the taskRetryLimit defined in the application profile before the task is failed.
- fail—When the method returns normally or with the specified control code, abort the session or fail the task, and propagate errors to the client application.

  For SessionEnter or SessionUpdate, immediately abort the session. Do not retry the method.

  For Invoke, immediately fail the task. Do not retry the method.
- succeed—This is a normal return. This is treated as a successful execution of SessionEnter, SessionUpdate, or Invoke.

  For Invoke, the taskcompletes successfully (in the Done state).

Not all values are possible with all methods. The table below outlines possible actions for methods:

Method	Possible value
Register	• No action possible
CreateService	• No action possible
SessionEnter	• retry • fail • succeed
SessionUpdate	• retry • fail • succeed
Invoke	• retry • fail • succeed
SessionLeave	• No action possible

Method	Possible value
DestroyService	• No action possible

Default values

Default values differ according to method.

Method	Default value
Register	• No action possible
CreateService	• No action possible
SessionEnter	• succeed
SessionUpdate	• succeed
Invoke	• succeed
SessionLeave	• No action possible
DestroyService	• No action possible

customizedDebugAction

Action to take on the service instance when the method returns normally or with the specified control code when service debugSetting="customized".

Where used

Service > Control > Method > Return

Required/Optional

Optional

Valid values

- writeServiceEventReplayFiles—When Symphony detects that the specified method has returned normally with or without a specific control code, it generates service event replay files.
- none—When Symphony detects that the specified method has returned normally with or without a specific control code, it does not generate service event replay files. This is the recommended setting, as this is typically a normal, successful event.

Default value

none

<Service ><Control><Method><Exception> element

Defines what action to take on service instances, sessions, or tasks when the method specified in <Method name=...> encounters a fatal or failure exception.

type

Exception type to which the configured action applies. Exception types cannot be configured for Register or DestroyService.

Where used

Service > Control > Method > Exception

Required/Optional

Required

Valid values

- failure— Exception type is FailureException.
- fatal— Exception type is FatalException.

Default values

No default value

controlCode

Numeric identifier that indicates to the system what action to take based on the number that is returned by the specified exception. There can be multiple control codes defined per exception, and different actions to take based on control code.

Where used

Service > Control > Method > Exception

Required/Optional

Required

Valid values

integers, positive and negative

Default value

0, default action taken on workload and service instances, as specified by default actions on actionOnSI and actionOnWorkload.

actionOnSI

Action to take on the service instance when a failure or fatal exception occurs.

Where used

Service > Control > Method > Exception

Required/Optional

Required

Valid values

- blockHost—When the specified exception occurs, terminate the running service instance on this host and do not use this host to start any other service instance for the application.

  The host on which the service instance was running is added to the blocked host list for the application. This host is no longer selected to run work for the application until it is explicitly unblocked through the EGO command line or the Platform Management Console.
- restartService—When the specified exception occurs, terminate the service instance, start a new service instance on the same host, and recover state. There is no limit to the number of times that a service instance can be restarted.
- keepAlive—When the specified exception occurs, take no action on the running service instance.

Note:

Actions cannot be configured for Register or DestroyService

Method	Possible values Fatal Exception	Possible values Failure Exception
Register	No action possible	No action possible
CreateService	• blockHost • restartService	• blockHost • restartService
SessionEnter	• blockHost • restartService • keepAlive	• blockHost • restartService • keepAlive
SessionUpdate	• blockHost • restartService • keepAlive	• blockHost • restartService • keepAlive
Invoke	• blockHost • restartService • keepAlive	• blockHost • restartService • keepAlive
SessionLeave	• blockHost • restartService • keepAlive	• blockHost • restartService • keepAlive
DestroyService	No action possible	No action possible

Default values

Method	Default values Fatal Exception	Default values Failure Exception
Register	No action possible	No action possible

Method	Default values Fatal Exception	Default values Failure Exception
CreateService	• blockHost	• blockHost
SessionEnter	• keepAlive	• keepAlive
SessionUpdate	• keepAlive	• keepAlive
Invoke	• keepAlive	• keepAlive
SessionLeave	• keepAlive	• keepAlive
DestroyService	No action possible	No action possible

actionOnWorkload

Action to take on sessions and tasks when a fatal or failure exception occurs.

Where used

Service > Control > Method > Exception

Required/Optional

Required

Valid values

- retry—When the specified exception occurs, retry the method up to the number of times configured by the session and task retry limits in the application profile.

  For SessionEnter and SessionUpdate, the system attempts to bind the session to the service instance up to the sessionRetryLimit in the application profile before the session is aborted.

  Note:

  The retry count for both of these methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3. Therefore the SessionRetryCount should be set to a value that accounts for both SessionEnter and SessionUpdate failures.

  For Invoke, the system attempts to run the task up to the taskRetryLimit defined in the application profile before the task is failed.
- fail—When the specified exception occurs, abort the session or fail the task, and propagate errors to the client application.

  For SessionEnter or SessionUpdate, immediately abort the session. Do not retry the method.

  For Invoke, immediately fail the task. Do not retry the method.

Not all values are possible with all methods. The table below outlines possible actions for methods:

Method	Possible values Fatal Exception	Possible values Failure Exception
Register	No action possible	No action possible
CreateService	No action possible	No action possible
SessionEnter	• retry • fail	• retry • fail
SessionUpdate	• retry • fail	• retry • fail
Invoke	• retry • fail	• retry • fail
SessionLeave	No action possible	No action possible
DestroyService	No action possible	No action possible

Default values

Default values differ according to method.

Method	Default values Fatal Exception	Default values Failure Exception
Register	No action possible	No action possible
CreateService	No action possible	No action possible
SessionEnter	• fail	• retry
SessionUpdate	• fail	• retry
Invoke	• fail	• retry
SessionLeave	No action possible	No action possible
DestroyService	No action possible	No action possible

customizedDebugAction

Action to take on sessions and tasks when a fatal or failure exception occurs in the specified method and debugSetting="customized".

Where used

Service > Control > Method > Exception

Required/Optional

Optional

Valid values

- writeServiceEventReplayFiles—When Symphony detects that the specified method has thrown a particular exception (FatalException or FailureException), as specified, it generates service event replay files to capture the relevant events that lead up to the exception.
- none—When Symphony detects that the specified method has thrown a particular exception (FatalException or FailureException), as specified, it does not generate service event replay files.

Default value

writeServiceEventReplayFiles

PART

III

Symphony 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
freeslot	available CPU slots	CPU slots			60 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 Linux/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 Linux/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 Linux/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.

Free slots (freeslot)

Current number of available slots for this host across ALL resource groups.

-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

Configuration Files

deployment.xml

The configuration file `deployment.xml` defines commands to be run after a service package is deployed. The file must be called `deployment.xml`, and must be included in the service package.

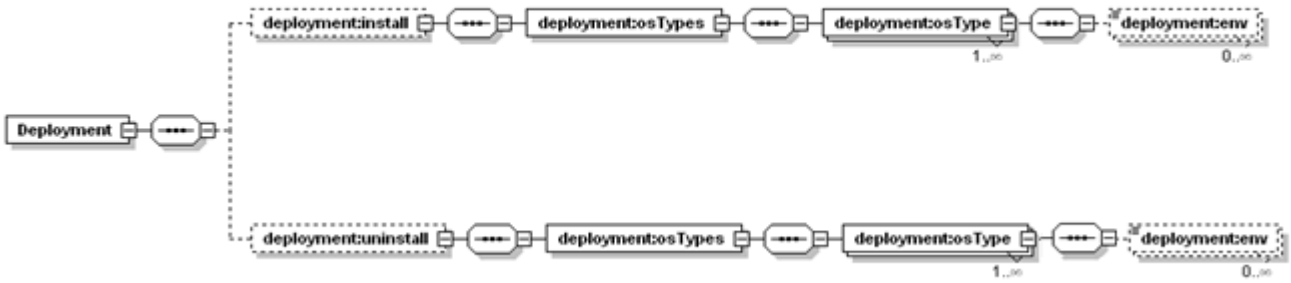
Location

The file must be included at the top level of the service package—it cannot be in a subdirectory. There can only be one file per service package.

Structure

`deployment.xml` has the following sections:

- install
- uninstall



Important:

All values in the `deployment.xml` file are case-sensitive when the service is deployed on Linux.

Example

```
<Deployment>
    <install>
        <osTypes>
            <osType name="NTX86" startCmd="setup" timeout="600" successCodes="0,1,2"/>
           <osType name="LINUX86" startCmd="install" timeout="100" successCodes="-1:10"/>
        </osTypes>
    </install>
    <uninstall>
        <osTypes>
            <osType name="NTX86" startCmd="setup -u" timeout="30" successCodes=0/>
            <osType name="LINUX86" startCmd="uninstall" timeout="34" successCodes="0"/>
        </osTypes>
    </uninstall>
</Deployment>
```

install section

Optional. Use the install section to configure commands to run after the package is uncompressed on a compute host.

osType attributes

Attribute	Description	Default Value
name	Required. Operating system type name. The name must match the osType indicated in the application profile, Service section. Default values: • NTX86—For a Windows-based environment • LINUX86—For a Linux-based environment	None

startCmd	Required.
	Command to run after the service package is copied on to a compute host and uncompressed.
	Specify a path relative to the service package installation directory.
	For example, if your package contained a subdirectory called scripts with the command you want to invoke called myscript, specify:
	Windows
	`.\scripts\myscript`
	Note:
	To run a Windows .bat script, you need to specify a special syntax. For example:
	`<osType name=NTX86" startCmd="cmd /c cmd /c install.bat" timeout="600" successCodes="0,1,2"/>`
	Linux
	`./scripts/myscript`
		None

timeout	Time that the startCmd is allowed to run before being terminated. Use this attribute to catch a runaway process. The time is counted from the moment the command specified in StartCmd is called.	600 seconds
successCodes	Return codes that indicate successful execution of the startCmd. Specify integers. To specify a list of codes, separate with commas. For example, `successCodes=”0,1”`. To specify a range of codes, separate with a colon. For example, to specify all codes from 0 to 10, enter `successCodes=”0:10”`. Note that if you specify success codes and the code returned is not amongst the specified successCodes, the system considers that the install startCmd failed and deployment is aborted.	Undefined. All return values indicate success.

uninstall section

Optional. Use the uninstall section to configure commands to run if the startCmd specified in the install section fails, or before the package is removed from a compute host.

osType attributes

Attribute	Description	Default Value
name	Required. Operating system type name. The name must match the osType indicated in the application profile, service section. Default values: • NTX86—For a Windows-based environment • LINUX86—For a Linux-based environment	None

startCmd	Required.
	Command to run if the startCmd specified in the install section fails, or before the package is removed from a compute host.
	For example, if your package contained a subdirectory called scripts with the command you want to invoke called myscript, specify:
	`.\scripts\myscript`
	Linux
	`./scripts/myscript`
		None

timeout	Time that the startCmd is allowed to run before being terminated. Use this attribute to catch a runaway process. The time is counted from the moment the command specified in StartCmd is called.	600 seconds
successCodes	Codes, which when returned, indicate successful execution of the startCmd. To specify a list of codes, separate with commas. For example, `successCodes="0,1"`. To specify a range of codes, separate with a colon. For example, to specify all codes from 0 to 10, enter `successCodes="0:10"`. Note that if you specify success codes and the code returned is not amongst the specified successCodes, the system considers that the uninstall startCmd failed. However, the package is still removed.	Undefined. All return values indicate success.

ego.conf (for Clients)

The configuration file used to connect to a Symphony cluster from a client machine that is not part of the cluster.

Location

This file is installed with Symphony DE and is located in `%SOAM_HOME%\conf\` on Windows, and in `$SOAM_HOME/conf/` on Linux.

Structure

You cannot change the file name `ego.conf`.

Note:

To connect to multiple clusters from the same client machine, configure different `ego.conf` files and rename them to `ego.conf` when you need to use them.

Parameters

EGO_MASTER_LIST

Syntax

```
EGO_MASTER_LIST=master_candidate1,master_candidate2 . . .
```

Description

Specifies the hosts that are master host candidates in the cluster to which you want to connect.

Valid values

The host names indicated here must exactly match the host names specified in the master `ego.conf`, accessible from the master host under `%EGO_CONFDIR%` on Windows and `$EGO_CONFDIR` on Linux.

Specify a list of hosts separated by commas.

EGO_KD_PORT

Syntax

```
EGO_KD_PORT=port_number
```

Description

Specifies the port number to use to connect to the Symphony cluster.

Valid values

The port number must exactly match the port number specified in the master `ego.conf` in the cluster.

Default

If the port is not specified, it defaults to 7870.

EGO_SEC_PLUGIN

Syntax

EGO_SEC_PLUGIN=sec_ego_default | *external_plugin*

Description

Specifies the security mechanism to use when connecting to the Symphony cluster.

Valid values

The value must exactly match the value specified in the actual `ego.conf` in the cluster. Specify one of the following:

sec_ego_default

Specifies to use the default authentication.

external_plugin

Specifies to use an external, third-party authentication.

Example

```
EGO_MASTER_LIST=hosta,hostc,hostd
EGO_KD_PORT=7870
EGO_SEC_PLUGIN=sec_ego_default
```

vem_resource.conf

Defines the hosts in the Symphony DE cluster. This file is not used with Symphony on the grid.

Location

The `vem_resource.conf` configuration file exists on each host in the SymphonyDE cluster, including the management host, and all compute hosts.

The default locations for `vem_resource.conf` are

- Windows—%SOAM_HOME%\conf
- Linux—$SOAM_HOME/conf

Structure

Each line defines a single host.

The `vem_resource.conf` file must be updated on each host whenever hosts are added or removed, or when the configuration parameters of any host in the network changes.

There are five types of hosts you configure: session manager, compute host, session director, repository service, and GUI service.

Session manager host:

AGENT: *port_number*:*host_name*:*max_SSMs*:0:*osType*:*CPU_factor*

Compute host:

AGENT: *port_number*:*host_name*:0:*max_SIMs*:*osType*:*CPU_factor*

Session director host:

SD_SDK: port_number:*host_name*:*sd_startCmd*

SD_ADMIN: *port_number*:*host_name*:*sd_startCmd*

Repository service host:

RS_DEPLOY: *port_number*:*host_name*:*rs_startCmd*

GUI service host:

WEBGUI: *port_number*:*host_name*:*startguiservice*

Attributes

port_number

For AGENT, specifies the port number the Platform EGO emulator (`start_agent`) uses. The `start_agent` process runs on each Symphony host.

For SD_SDK, SD_ADMIN, specifies port numbers used by the session director.

For RS_DEPLOY, specifies the port number used by the repository service.

For WEBGUI, specifies the port number used by the GUI service. Port number for WEBGUI is 18080.

host_name

Specifies localhost in single-host environment.

In a multi-host environment, specify using dotted-decimal notation or as a DNS name. Hosts can also have dynamic IP addresses (DHCP).

max_SSMs

Specifies the maximum number of session managers that can be started on this host. For a compute host, specify **0**.

max_SIMs

Specifies the maximum number of service instance managers that can be started on this host. Each service instance manager requires one (virtual) slot to run on, therefor the maximum number of sims the compute host can run is equal to the number of CPU slots on the machine. For a dedicated session manager host, specify **0** to prevent work from running on this host.

osType

Specifies the operating system type the host runs:

- LINUX86—A Linux-based environment, such as RedHat Linux
- NTX86—A Windows-based environment, such as Windows 2003, Windows 2000, Windows NT, and Windows XP

CPU_factor

Default=1.

The CPU factor is 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. CPU factors are defined by the cluster administrator. For multiprocessor hosts, the CPU factor is the speed of a single processor.

sd_startCmd

Specifies the command that starts the session director.

Specify `sd` unless you have changed the name of the command or moved the session director executable from the installation directory. In this case, specify the absolute path to the session director executable.

rs_startCmd

Specifies the name of the command that starts the repository service.

Specify `rs` unless you have changed the name of the command or moved the respository service executable from the installation directory. In this case, specify the absolute path to the repository director executable.

GUI_startcmd

Specifies the name of the command that starts the WebGUI service.

Specify `startguiservice` unless you have changed the name of the command or moved the startguiservice script from the installation directory. In this case, specify the absolute path to the startguiservice script.

Examples

Management host also runs workload

In the following example, host1 runs the session director, repository service, GUI service, five session manager processes, and five service instance manager processes per application.

```
# Resource configuration file
  #
  # File format:
  # <service name>:<port_number>:<host_name>:<max number of
SSMs SD can start>:<max number of SIMs SSM can
start>:<OS_type>:<CPU_factor>
  #
  AGENT:8000:host1:5:5:LINUX86:1
  #
  # SD service information
  # <service name>:<port_number>:<host_name>:<sd startcmd>
   SD_SDK:15051:host1:sd
   SD_ADMIN:15050:host1:sd
   #
  # RS service information
  # <service name>:<port_number>:<host_name>:<rs startcmd>
   RS_DEPLOY:15052:host1:rs
 #
# GUI service information
  # <service name>:<port_number>:<host_name>:<gui startcmd>
# Note:The port number of WEBGUI is fixed as 18080.
   WEBGUI:18080:host1:startguiservice
```

One management host and one compute host

In the following example, the host `myfirsthost` runs the session director, the repository service, five session manager processes and GUI service. The compute host, `mysecondhost` runs five service instance manager processes per application.

```
# Resource configuration file
  #
  # File format:
  # <service name>:<port_number>:<host_name>:<max number of
SSMs SD can start>:<max number of SIMs SSM can
start>:<OS_type>:<CPU_factor>
  #
  AGENT:8000:myfirsthost:5:0:LINUX86:1
  AGENT:8000:mysecondhost:0:5:LINUX86:1
  #
  # SD service information
  # <service name>:<port_number>:<host_name>:<sd startcmd>
   SD_SDK:15051:myfirsthost:sd
   SD_ADMIN:15050:myfirsthost:sd
   #
  # RS service information
  # <service name>:<port_number>:<host_name>:<rs startcmd>
   RS_DEPLOY:15052:myfirsthost:rs
  # WebGUI service information
  # <service name>:<port_number>:<host_name>:<gui startcmd>
# Note:The port number of WEBGUI is fixed as 18080.
   WEBGUI:18080:myfirsthost:startguiservice
```

ego.sudoers

Contents

- About ego.sudoers
- The ego.sudoers file
- File format
- Creating and modifying ego.sudoers
- Parameters

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.

PART

V

Environment Variables

Symphony client environment variables

This topic describes the environment variables that the Symphony client uses.

API_CALL_TIMEOUT

Syntax

```
API_CALL_TIMEOUT=value
```

Description

Defines the number of seconds the Symphony API waits to complete a connection to the session director or session manager before it times out.

For a client to submit workload, the client must connect to an application and then interact with a session created on this connection.

The timeout value can be any unsigned integer and -1, which is treated specially to specify that the API never times out.

Do not specify zero (0) because then the API never waits and so never connects.

Default

-1 (never times out)

RS_REQUEST_TIMEOUT

Syntax

```
RS_REQUEST_TIMEOUT=value
```

Description

Specifies the number of seconds the deployment command waits for the Repository Server (RS) to send or receive a service package to or from the RS before it times out.

The timeout value can be any positive integer and -1, which specifies that the deployment command never times out.

If you specify zero (0), the default value of 300 seconds is applied.

Default

300 (seconds)

Client reconnection environment variables

The following are the four environment variables that you can define on your client machine to influence how the client behaves when it loses its connection to the session manager:

- SOAM_RECONNECTION_RETRY_INTERVAL
- SOAM_RECONNECTION_RETRY_LIMIT
- SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL
- SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT

How variables work together

When a client loses its connection to the session manager, it tries to reconnect to the existing session manager up to the SOAM_RECONNECTION_RETRY_LIMIT. Each reconnection attempt is SOAM_RECONNECTION_RETRY_INTERVAL seconds apart.

- If the client succeeds in reconnecting to a session manager, the API attempts to refresh all sessions on the connection. If the session manager was relocated during the disconnection, any non-recoverable sessions are aborted.
- If reconnection was unsuccessful:

  The API stops trying to reconnect to the existing session manager and attempts to locate a new session manager. It then tries to locate and connect to a new session manager up to the SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT. Each reconnection attempt is SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL seconds apart.

  - If the client fails to reconnect to a session manager, the API stops trying to reconnect and throws an exception indicating that Symphony is no longer available.
  - If the client succeeds in reconnecting to a session manager, the API attempts to refresh all sessions on the connection. If the session manager was relocated during the disconnection, any non-recoverable sessions are aborted.

Important:

Compatibility Note: The variables CONNECTION_RETRY_INTERVAL and CONNECTION_RETRY_LIMIT have been deprecated as of Symphony 4.0. This means that they continue to work in Symphony 4.1 but you are encouraged to use the variables that replace them. If you use the deprecated variables, all previous variable values are preserved. If you use any of the new variables, any definition of the deprecated variables is ignored and all new variable defaults are used. Mixing the deprecated variables with the newer variables always results in the newer variables and their default values (if not specified) being applied.

CONNECTION_RETRY_INTERVAL

Syntax

CONNECTION_RETRY_INTERVAL=*value*

Description

This variable is deprecated as of Symphony 4.0 and is replaced by SOAM_RECONNECTION_RETRY_INTERVAL and SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL. If you used this variable with previous versions of Symphony, it still works the way it used to.

If you need to influence the tolerance of a broken connection between your client and the system, use the following variables:

- SOAM_RECONNECTION_RETRY_INTERVAL
- SOAM_RECONNECTION_RETRY_LIMIT

- SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL
- SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT

Default

3 (seconds)

CONNECTION_RETRY_LIMIT

Syntax

CONNECTION_RETRY_LIMIT=*value*

Description

This variable has been deprecated as of Symphony 4.0 and has been replaced by SOAM_RECONNECTION_RETRY_LIMIT and SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT. If you have used this variable with previous versions of Symphony it will still work in the way it used to.

If you need to influence the tolerance of a broken connection between your client and the system, use the following variables:

- SOAM_RECONNECTION_RETRY_INTERVAL
- SOAM_RECONNECTION_RETRY_LIMIT
- SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL
- SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT

Default

5 (times)

SOAM_RECONNECTION_RETRY_INTERVAL

Syntax

SOAM_RECONNECTION_RETRY_INTERVAL=*value*

Description

Specifies how long the API waits, in seconds, before the next attempt to reconnect to the last known instance of the session manager.This value is used in conjunction with SOAM_RECONNECTION_RETRY_LIMIT.

Note:

On Windows, the effective minimum interval is 1 second, even if you set the value to 0.

Default

1 (second)

SOAM_RECONNECTION_RETRY_LIMIT

Syntax

SOAM_RECONNECTION_RETRY_LIMIT=*value*

Description

Specifies the number of attempts made by the API to reconnect to the last known instance of the session manager.This variable is used to handle the case of a transient disconnection from the system for any type of session. When a disconnection is sensed by the API, instead of throwing an exception for a pending operation, the API attempts to automatically reconnect to the last known instance of the session manager for the specified number of times. When the API has reached the number of attempts without reconnection, it attempts to connect to a new instance of the session manager and uses SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT.

Default

15 (times)

SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT

Syntax

SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT=*value*

Description

Specifies the number of attempts made by the API to reconnect to a new instance of the session manager.

This variable is used to handle the case in which session manager is relocated (either restarted on the same host or failed over to a different host). This variable is used by the API to locate a new instances of the session manager after SOAM_RECONNECTION_RETRY_LIMIT is reached.

Note:

On Windows, the effective minimum interval is 1 second, even if you set the value to 0.

Default

60 (times)

SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL

Syntax

SOAM_RELOCATED_RECONNECTION_RETRY_INTERVAL=*value*

Description

Specifies how long the API waits, in seconds, before the next attempt to reconnect to the new instance of the session manager.This value is used in conjunction with SOAM_RELOCATED_RECONNECTION_RETRY_LIMIT.

Default

5 (seconds)

TCP connection environment variables

You can set the following environment variables on your client machine to define the characteristics of the TCP connection:

- PLATCOMMDRV_TCP_KEEPALIVE_TIME
- PLATCOMMDRV_TCP_NODELAY
- PLATCOMMDRV_TCP_RECV_BUFFER_SIZE
- PLATCOMMDRV_TCP_SEND_BUFFER_SIZE

PLATCOMMDRV_TCP_KEEPALIVE_TIME

Syntax

PLATCOMMDRV_TCP_KEEPALIVE_TIME=*seconds*

Description

Specifies the maximum time in seconds the system can take to detect a broken connection to and from Symphony processes originating from the shell where the variable is defined.

Specify a numeric value of 180 seconds (3 minutes) or more. Values lower than 300 seconds (5 minutes) are not recommended.

Restriction:

This variable is ignored for installations on Solaris, which does not support this TCP/IP setting on a per-socket basis.

Default

Connections by default use the operating system-dependable, system-wide TCP keep alive setting, which for most operating systems, is 7200 seconds, or 2 hours.

PLATCOMMDRV_TCP_NODELAY

Syntax

PLATCOMMDRV_TCP_NODELAY=`1 | 0`

Description

Enables or disables the Nagle algorithm for TCP sockets.

Specify 1 to enable, 0 to disable.

Default

1 (enabled).

PLATCOMMDRV_TCP_RECV_BUFFER_SIZE

Syntax

PLATCOMMDRV_TCP_RECV_BUFFER_SIZE=*bytes*

Description

Specifies the size for the TCP receive buffer for each Symphony connection.

Specify a numeric value greater than or equal to 65535 bytes (64KB).

Default

65535 (64KB).

PLATCOMMDRV_TCP_SEND_BUFFER_SIZE

Syntax

PLATCOMMDRV_TCP_SEND_BUFFER_SIZE=*bytes*

Description

Specifies the size for the TCP send buffer for each Symphony connection.

Specify a numeric value greater than or equal to 65535 bytes (64KB).

Default

65535 (64KB).

System-defined environment variables

Environment variable	Description
SOAM_BINDIR	Directory where the Symphony middleware commands are installed
SOAM_HOME	Directory where the Symphony middleware is installed
SOAM_SERVER_DIR	Directory where the Symphony middleware daemons are installed

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_CLIENT_ADDR	Used with firewalls and rfa. Range of listening ports for clients to connect back to rfa. This environment is useful if you have a firewall configured on the client side. Example: EGO_CLIENT_ADDR=56000-56020	There is no default value. If not set, the client will listen on a random port.
EGO_CONFDIR	The directory where the valid EGO configuration file ego.conf is stored. The file may be duplicated in the system, but the cluster only uses the file stored in this location.	On Linux: *EGO_TOP*/kernel/conf or *shared_dir*/kernel/conf On Windows: *EGO_TOP*\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

Environment variable	Description	Default Value
EGO_LOCAL_CONFDIR	This is the local configuration directory.	On Linux: *EGO_TOP*/kernel/conf On Windows: *EGO_TOP*\kernel\conf
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.

Index

A

aborted session state 102
abortSessionIfClientDisconnect in application profile 168
abortSessionIfTaskFail in application profile 168
activities
    viewing 24
activity view subcommand 24
administrators
    removing 45
alloc list subcommand 24
alloc modify subcommand 25
alloc new subcommand 25
alloc release subcommand 26
alloc unblock subcommand 27
alloc view subcommand 27
allocation free subcommand 24
allocations
    displaying for client 24
    displaying for consumer 24
    reducing number of slots 26
    releasing 29
    removing for client 24
    removing for consumer 24
    requesting more slots 25
    requesting new 25
    unblocking a host 27
    viewing 27
API methods
    defining time out 225
API_CALL_TIMEOUT environment variable 225
app -c in soamview 100
app disable in soamcontrol 62
app enable in soamcontrol 61
app in soamview 100
application profile
    deploymentTimeout 173
    how it is removed 97
    updating 87
    viewing 99
application profile attributes
    abortSessionIfClientDisconnect 168
    abortSessionIfTaskFail 168
    applicationName 123
    consumerId 123
    description 171, 173, 174
    discardResultsOnDelivery 170
    diskSpace 152, 153, 155, 157, 159
    elementName 144
    fileSwitchSize 149
    fileSwitchTime 150
    lastingPeriod 150, 176
    name (event) 146
    name (Service) 171
    numOfPreloadedServices 134
    path 151
    persistSessionHistory 170
    persistTaskHistory 169, 170
    policy 128
    pollFrequency 148
    preExecCmd 179
    preStartApplication 133
    reclaimGracePeriod 166
    recoverable 164
    resourceBalanceInterval 132
    resourceGroupName 124, 142
    resReq 124, 138
    sessionSchedulingInterval 83, 133
    shutDownTimeout 138
    suspendGracePeriod 167
    taskCleanupPeriod 167
    taskGracePeriod 166
    taskLowWaterMark 132
    taskRetryLimit 165
    transientDisconnectionTimeout 134, 135, 168
    value 146
applicationName in application profile 123
applications

controlling 61
disabling 62
effects of unregistering an application 97
enabling 61
logging session manager messages 76
logging workload messages 75
naming 123
prioritizing sessions 83
registering 87
releasing resources 97
saving historic data 97
starting soamview counters 99
states
disabled and enabled 101
unregistering 97
updating 87
updating application profile 88
viewing disabled applications 100
viewing enabled applications 100

B

BEV_CRITICAL boundary event 147
BEV_HALT boundary event 147
BEV_PROACTIVE boundary event 147
BEV_SEVERE boundary event 147
blockHostOnTimeout parameter, application profile 147
boundary events 145
monitoring 147
monitoring memory usage 146
system responses 145
triggering system responses 145
boundary manager
monitoring virtual address space 144
monitoring virtual memory 144

C

client list subcommand 28
client reg subcommand 28
client unreg subcommand 29
client view subcommand 29
clients
defining behavior when disconnected 168
descreasing response time 133
generating list of 28
optimizing performance 133
problem running
EGO_CONFDIR environment variable 232
recovering from lost connections 225
registering 28
unregistering 29
viewing information about 29
closed session state 102
cluster
name
displaying 34
restarting 34
starting 35
starting all components 48
stopping 35
stopping all components 47
cluster administrators
listing 45
clusters
changing priority of all sessions 83
defining hosts 216
command line
logging off 44
logging on 44
quitting 36
commands
egoconfig 11
egoconfig join 11
egoconfig masterlist 11
egoconfig mghost
on UNIX 11
on Windows 12
egoconfig mghost soam
on UNIX 12
on Windows 13
egoconfig unsetmghost 14
egoremoverc 15
egosetrc 16
egosetsudoers 17
egosh activity view 24
egosh alloc list 24
egosh alloc modify 25
egosh alloc new 25
egosh alloc release 26
egosh alloc unblock 27
egosh alloc view 27
egosh allocation free 24
egosh client list 28
egosh client reg 28
egosh client unreg 29

egosh client view 29
egosh consumer alloc 29
egosh consumer list 29
egosh consumer view 30
egosh debug pemoff 30, 31
egosh debug pemon 30, 31
egosh debug vemkdoff 32
egosh debug vemkdon 32
egosh ego elimrestart 33
egosh ego info 34
egosh ego restart 34
egosh ego shutdown 35
egosh ego start 35
egosh exit 36
egosh quit 36
egosh resource close 36
egosh resource group 37
egosh resource list 38
egosh resource list -R 39
egosh resource open 40
egosh resource setpriority 40
egosh service list 41
egosh service start 41
egosh service stop 41
egosh service view 41
egosh user add 42
egosh user assignrole 42
egosh user delete 43
egosh user execpasswd 33
egosh user list 43
egosh user logoff 44
egosh user logon 44
egosh user modify 44
egosh user roles4user 45
egosh user unassignrole 45
egosh user users4role 45
egosh user view 46
egoshutdown 47
egostartup 48
prestartApplication 137
preStartApplication 134
pversions 49
rsdeploy 50, 55
running in a batch file 179
running in a script 179
soamcontrol
app disable subcommand 62
app enable subcommand 61
session kill subcommand 63, 64
session resume subcommand 67, 68
session suspend subcommand 65, 66
soamdeploy
remove subcommand 72, 95
view subcommand 73
soamlog
sd subcommand 75
sim subcommand 78, 79
ssm subcommand 76
workload subcommand 80
soamlogoff 81
soamlogon 82
soammod session 83
soamreg 87
soamshutdown 91
soamstartup 92
soamunreg 97, 106
soamview
app subcommand 100
session subcommand 101, 103
task subcommand 103, 105
specifying
session password 82
session user name 82
common data
optimizing paging 156, 158
specifying blocksize 156, 158
common data updates 145
compute host
configuring number of CPU slots 217
compute hosts
allocating for consumer 124
allocating for session manager 138
defining 216
deploying packages 70
processes 91
shutting down 91
starting up 92
compute resources
CPU slots 217
configuration
adding local host to cluster 11
automatic startup
removing 15
configuration master candidates list 11
demote management host to compute host 14
host members 216

locating files 216
move to shared configuration file from local 11–13
root privileges 17
configuration files
soam.conf 214
vem_resource.conf 92, 216
consumer administrators
listing 45
consumer list subcommand 29
consumer users
listing 45
consumer view subcommand 30
consumerId in application profile 123
consumers
allocation and demand information, summary 29
listing for a client 29
listing for a host 29
listing for allocation ID 29
summary of allocation and demand information 29
viewing information about 30
control
controlling tasks, sessions, and applications 61
counters
archive period 150
polling frequency 148
session manager shutdown period 138
switching files 149
task grace period 166, 167
transientDisconnectionTimeout 134
CPU
factors
static resource 126, 140, 217
utilization
ut load index 202
CPU factor
description 199
CPU factor (cpuf) static resource 126, 140
CPU slots
configuring 217
CPU utilization
description 199
CPU_factor in vem_resource.conf 217
cpuf static resource 126, 140, 217
CPUs
number of
description 199
specifying 217

D

data 144
archived history files 150
common data
scheduling policy for data affinity 129
history files 150
persisting task history 169
specifying maximum age of history file 149
specifying maximum history file size 149
triggering file archival 149
tuning
monitoring available memory 144
triggering memory low conditions 145
triggering memory low events 146
data files
specifying path
history 154, 156, 158
journaling sessions 154, 156, 158
journaling tasks 154, 156, 158
paging common data 154, 156, 158
paging tasks 154, 156, 158
storing 154, 156, 158
debug
of pem
turning off 30, 31
turning on 30, 31
of vemkd
turning off 32
turning on 32
debug information
service instance manager 78, 79
workload logging levels 76–79
debug log levels
workload 76–79
debug pemoff subcommand 30, 31
debug pemon subcommand 30, 31
debug vemkdoff subcommand 32
debug vemkdon subcommand 32
deploymentTimeout 173
description in application profile 171, 173, 174
directories
for binaries 231
for EGO libraries 231
for EGO services 231
for ego.conf 231
for server binaries 232
disabled application state 101
disk space
defining maximum size 153, 155, 157, 159

for journaling 153, 155, 157, 159
for paging 153, 155, 157, 159
disks
description 200
I/O rate 202
diskSpace in application profile 152, 155, 157, 159

E

effective run queue length 201
ego info subcommand 34
ego restart subcommand 34
ego shutdown subcommand 35
ego start subcommand 35
EGO_BINDIR environment variable 231
EGO_CONFDIR environment variable 231
EGO_ESRVDIR environment variable 231
EGO_LIBDIR environment variable 231
EGO_SERVERDIR environment variable 232
EGO_STARTUP_ALTERNATE_PATHS
ego.sudoers file 220
EGO_STARTUP_PATH
ego.sudoers file 220
EGO_STARTUP_USERS
ego.sudoers file 221
ego.conf file
directory where stored 231
ego.sudoers file 219
creating 17
egoconfig command 11
egoremoverc command 15
egosetrc command 16
egosetsudoers command 17
egosh command
getting help 18
egoshutdown command 47
egostartup command 48
elementName in application profile 144
enabled application state 101
environment variables
API_CALL_TIMEOUT 225
defining path for preExec commands 179
EGO_BINDIR 231
EGO_CONFDIR 231
EGO_ESRVDIR 231
EGO_LIBDIR 231
EGO_SERVERDIR 232
list of 231
errors
changing details logged 75
events
boundaries 147
memory boundaries 145
naming 145
execution priority 126, 140
execution sessions
closing 106
create session, run command and closing session at once 106
creating 106
exit subcommand 36

F

failover
facilitating 154, 156, 158
recovering sessions 164
files
determining when to archive 148
facilitating recovery 143
monitoring frequency 148
polling frequency 148
sd.log4j.properties 75
sim.log4j.properties 78, 79
SOAM_HOME/conf/vem_resource.conf 216
specifying location of working files 143
storing on shared file system 143
vem_resource.conf 216
working and temporary 143
fileSwitchSize in application profile 149
fileSwitchTime in application profile 150
free memory 202
free slots
number of
description 199

G

grace periods
overriding 91
specifying for suspended sessions 166, 167
specifying for task cleanup 167

H

historic data
saving 97
history files

archiving 149
deleting archived files 150
logging historical information 149
polling 149
renaming 149
retaining archived files 150
specifying archive retention period 150
specifying maximum size 149
specifying retention time for archived files 150
hname static resource 126, 140
host model
description 199
host model static resource 126, 140
host name static resource 126, 140
host names
description 199
host properties
1-minute load 199
15-minute load 199
15-second load 199
CPU factor 199
CPU util 199
descriptions 199
disks 200
host model 199
host name 199
host status 199
host type 199
I/O rate 199
idle time (It) 200
max swap 200
Max Temp 200
maximum memory 199
mem 199
number of CPUs 199
number of free slots 199
number of slots 199
paging rate (pg) 199
swap 199
temp 200
users 200
host scavenging subcommand
ego elimrestart 33
host states 199
host type
description 199
host type static resource 126, 140
host types
displaying list of 38
host_name in vem_resource.conf 217
hosts
adding local to cluster 11
changing log levels 75
closing 36
configuring master candidates list 11
defining 216
displaying groups of 37
opening 40
set process priority 40
shutting down 91
shutting down a management host 91
shutting down all 91
shutting down local 91
starting up 92

I

I/O rate
description 199
idle time
built-in load index 202
idle time (It)
description 200
install_dir/conf/vem_resource.conf 216
io load index 202
it load index
description 202

J

join subcommand 11

L

lastingPeriod in application profile 150, 176
list
of user accounts
displaying 43
load
1-minute load
description 199
15-minute load
description 199
15-second load
description 199
load average 201

load indices
io 202
it 202
ls 202
mem 202
pg 202
r15m 201
r15s 201
r1m 201
swp 202
tmp 202
ut 202
log files
effects of closing applications 75
effects of restarting 75
log levels
changing 75
effecting changes 75
workload 76–79
logging levels
setting to LOG_TRACE 31, 32
login sessions 202
login users (users)
description 200
ls load index 202

M

management hosts
configuring 216
processes
rs 91
sd 91
ssm 91
start_agent 91
shutting down 91
starting 92
master candidates
displaying list of 38
master host
displaying name of 34
masterlist subcommand 11
max _SSMs in vem_resource.conf 217
max swap
description 200
Max Temp
description 200
max_SIMs in vem_resource.conf 217
maximum memory
description
hosts 199
maxmem static resource 126, 140
maxswp static resource 126, 140
maxtmp static resource 126, 140
mem load index 202
memory
available 202
description 199
memory low conditions
causes of 145
correcting 146
messages
changing details logged 75
effects of message blocksize 145
mghost soam subcommand 12, 13
mghost subcommand 11, 12
middleware packages
deploymnet and removal 50, 55
model static resource 126, 140

N

name
application profile attributes
osTypes 175
Service 171
name (event) in application profile 146
name (Service) in application profile 171
ncpus static resource 126, 140
ndisks static resource 126, 140
normalized run queue length
description 201

O

open session state 102
operating systems
naming relevant 175
optimization
client response time 133, 134
preloading services 134
prestarting applications 133
order string 128, 142, 206
osType in vem_resource.conf 217

P

packages
  deploying 70
  uncompressing 70
paging block size 152, 154
paging rate
  description 202
  load index 202
paging rate (pg)
  description 199
passwords
  changing 44
    for Windows user account
      setting on Linux 33
patches
  viewing installed on Windows 49
path in application profile 151
pem
  debugging
    turning off 30, 31
persistSessionHistory in application profile 170
persistTaskHistory in application profile 170
Platform EGO emulator 216
policy in application profile 128
pollFrequency in application profile 148
port_number in vem_resource.conf 216
ports for EGO emulator 216
preExecCmd in application profile 179
preStartApplication in application profile 133
priority
  modifying 83
privileges
  removing 45
processes
  logging service instance manager 75
  logging session director 75
  logging session manager 75
  Platform EGO emulator 91, 216
  prestarting 133
  reasons for limiting on session manager 146
  repository service 91
  service instance 91
  service instance manager 91
  session director 91
  session manager 91
  shutting down 91
  start_agent 91, 216
  starting at workload request 92
  starting on management hosts 92
profile
  version of 122
properties files
  sd.log4j.properties 75–79
  sim.log4j.properties 78, 79
  ssm.log4j.properties 76
pversions command 49

Q

quit subcommand 36

R

r15m load index
  built-in resources 201
r15s load index
  built-in resources 201
r1m load index
  built-in resources 201
reclaimGracePeriod in application profile 166
recoverable in application profile 164
recovery
  defining lost connection behavior 225
  defining path to journaling files 154, 156, 158
  saving session information 164
  transientDisconnectionTimeout 134
  waiting for client reconnection 134
remote jobs
  execution priority 126, 140
remove in soamdeploy 72, 95
resource allocation
  balancing 132
  compute resources 124
  consumer 124, 142
  session manager 138
  triggering resource release 132
    tuning
      balance frequency 132
resource close subcommand 36
resource group subcommand 37
resource groups
  displaying information about 37
resource list -R subcommand 39
resource list subcommand 38
resource open subcommand 40
resource requirement string 125, 139, 203

resource requirements
operators 126, 140, 204
sorting 128, 142, 206
resource setpriority subcommand 40
resourceBalanceInterval in application profile 132
resourceGroupName in application profile 124
resources
balancing 132
balancing interval 132
closing 36
CPU factor 217
CPU slots 217
displaying
information about 40
list that matches string 39
master candidates 38
displaying allocations for 24
displaying demand 37
displaying resource shares 37
freeing for client 24
freeing for consumer 24
grouping 125, 139, 203
monitoring boundary values 146
monitoring events 146
monitoring memory 144
opening 40
polling frequency 132
reallocating session compute resources 133
reducing request 26
releasing 26, 97, 132
requesting 25, 132
requesting additional 25
scheduling sessions 133
setting OS process priority 40
sorting 128, 142, 206
specifying 125, 139, 203
viewing allocations of 27
rexpri static resource 126, 140
roles
for users
assigning 42
root permissions
granting 17
rs process 91
rsdeploy 55
rsdeploy command 50, 55
run queue
effective 201
normalized 201

S

scheduling policies
R_MinimumServices 129
scheduling policies, R_Proportion 129
schema
version of 122
sd in soamlog 75
sd process 91
sd.log4j.properties 75–79
security
automatically logging off 82
selection strings
operators 126, 140, 204
server static resource 126, 140
service execution
aborting sessions 168
custom service deployment 179
defining termination cleanup period 167
deploying a service 179
killing a session 167
specifying reclaim grace period, tasks 166
specifying suspension grace period 166, 167
service instance manager
changing log levels 75
prestarting 133
service instance manager, max number of SIMs 217
service instances
prestarting 133, 134
specifying number to prestart 134
starting 134
service list subcommand 41
service start subcommand 41
service stop subcommand 41
service view subcommand 41
serviceName 162
services
copying to the local host 179
displaying information about 41
listing 41
specifying
name 171
starting 41
stopping 41
viewing deployed service packages 73
session director

changing log levels 75
session in soamview 103
session kill, soamcontrol subcommand 63
session manager
changing log levels 75
limiting processes run by 146
locating executables 175
monitoring memory usage 144, 145
prestarting 133
shutting down 138
specifying client reconnection period 134
specifying max for host 217
specifying shutdown timeout 138
session resources
increasing and decreasing 83
session resume in soamcontrol 67, 68
session suspend in soamcontrol 65, 66
session, in soammod 83, 85
sessions
aborting 168
applying priority changes 83
changing priority 83
closing after task cleanup period 167
closing after task grace period 166, 167
controlling 61
default Symphony session length 82
defining when to abort 168
duration 82
expiring time to live 82
killing 64
modifying priority 83
overriding logon user 82
priority 83
priority number range 84, 85
reasons for terminating 146
recovering 164
resuming 67
retaining information 164
setting time to live 82
specifying highest priority 84, 85
specifying scheduling interval 133
specifying Symphony session length 82
states
aborted 102
closed 102
open 102
suspended 102
suspending 65, 66
task cleanup period 167
task grace period 166, 167
terminated by soamreg 88
terminated by soamunreg 97
terminating if task fails 168
terminating with soamcontrol 63, 64
viewing active sessions 101
viewing information 99
viewing running sessions 101
viewing suspended sessions 101
sessionSchedulingInterval in application profile 133
shutDownTimeout in application profile 138
sim in soamlog 78, 79
sim.log4j.properties 78, 79
slots
number of
description 199
releasing 26
specifying 217
soamcontrol command 61
soamdeploy command 70
soamlog command 75
soamlogoff command 81
soamlogon command 82
soammod command 83
soamreg command 87
soamshutdown command 91
soamstartup command 92
soamunreg command 97, 106
soamview command 99
ssm in soamlog 76
ssm process 91
ssm.log4j.properties 76
start_agent in vem_resource.conf 216
start_agent process
shutting down 91
specifying ports 216
starting up 92
states
viewing canceled tasks 103
viewing done tasks 103
viewing pending tasks 103
viewing running tasks 103
static resources
See also individual resource names
description 125, 139
suspended session state 102
suspendGracePeriod in application profile 167

swap
description 199
swap space
load index 202
swp load index
description 202

T

task history
rerun 169
rerun counts 169
runtime 169
task in soamview 103
task, soamview subcommand 105
taskGracePeriod in application profile 166
taskLowWaterMark in application profile 132
taskRetryLimit in application profile 165
tasks
aborting after task cleanup period 167
accommodating large paging files 152, 154
canceling after task grace period 166, 167
clean up duration 167
continuing to run after suspension 166, 167
CPU slot ratio 132
efficiently using resources 132
paging block size 152, 154
releasing resources 132
retaining history 169
specifying low-water mark 132
states
canceled 103
done 103
pending 103
running 103
suspending 166, 167
terminated with soamunreg 97
viewing by task state 103
viewing closed 103
viewing information 99, 105
temp
description 200
time to live
specifying for a session 82
tmp load index
description 202
transientDisconnectionTimeout in application profile 134, 135
tuning
available memory 144
monitoring virtual address space 144
type static resource 126, 140

U

unsetmghost subcommand 14
user accounts
assigning role to 42
changing 44
creating 42
deleting 43
listing 43
listing by role 45
listing roles for 45
removing role from 45
user add subcommand 42
user assignrole subcommand 42
user delete subcommand 43
user execpasswd subcommand 33
user list subcommand 43
user logoff subcommand 44
user logon subcommand 44
user modify subcommand 44
user roles
assigning 42
listing by user 45
listing users 45
removing 45
user roles4user subcommand 45
user unassignrole subcommand 45
user users4role subcommand 45
user view subcommand 46
users
assigning roles to 42
changing session priority 83
displaying information about 46
logging off 81
logging on 82
specifying password 82
specifying session length 82
ut load index
built-in resource 202

V

value in application profile 146

vem_resource.conf
configuration file 216
specifying attributes 217
vem_resource.conf attributes
host_name 217
port_number 216
vemkd
turning off debug 32
turning on debug 32
version
displaying 34
of EGO
displaying 18
view in soamdeploy 73
virtual address space
monitoring 145
virtual memory
defined 144
load index 202
monitoring 144

W

Windows
user accounts
setting password for on Linux 33
working files
storing on shared file system 154, 156, 158
workload
changing log levels 75
workloadapi in soamlog 80