Cluster and Application Management Guide

Platform Symphony
Version 5.1
April 2011

Platform
Computing

Copyright

We’d like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Document redistribution and translation

This document is protected by copyright and you may not redistribute or translate it into another language, in part or in whole.

Internal redistribution

You may only redistribute this document internally within your organization (for example, on an intranet) provided that you continue to check the Platform Web site for updates and update your version of the documentation. You may not make it available to your organization over the Internet.

Trademarks

Third-party license agreements

http://www.platform.com/Company/third.part.license.htm

Third-party copyright notices

http://www.platform.com/Company/Third.Party.Copyright.htm

Contents

About this guide

This guide describes how to manage and configure your cluster, and deploy and manage Symphony applications.

This guide uses extensive references to Linux/UNIX. In these cases, the same information is applicable to both Linux and UNIX operating systems. In the context of this guide, Solaris is considered a UNIX operating system. Where differences occur, operating system-specific information is provided.

I

Cluster Management

CHAPTER

1

Controlling EGO

Start EGO on the Windows cluster

You must have administrator access on the local Windows host.

1. If this is a Windows cluster, start EGO on all hosts in the cluster:

   egosh ego start all

   This starts all Windows hosts.
2. If this is a mixed cluster, start EGO on all Windows and all Linux/UNIX hosts in this way:
   a) Log on to a Windows host in the cluster and run **egosh ego start all**.
   b) Log on to a Linux/UNIX host in the cluster, set the environment variables, and then run **egosh ego start all**.

Note:

You can use any local administrator account to start EGO. If the current logon user is not Administrator, you will need to input the password of the Administrator account every time you start EGO. To start the cluster, use the following command: runas /user:*Admin_account_name* "egosh ego start all".

Start EGO on the Linux/UNIX cluster

Log on with root permissions. Ensure rsh is available on each host in the cluster.

Important:

By default, only root can start, stop, or restart the cluster. Optionally, you can grant root privileges to egoadmin, the cluster administrator account.

---

1. Start EGO on all Linux/UNIX hosts in the cluster:

   ```
   egosh ego start all
   ```

Shut down EGO on the Windows cluster

You must be logged on to any host in the cluster as the local system user and have EGO administrative privileges.

Note:

If the master is down, log on to a management host (if the master is down, there is no way to get the complete list of cluster services and daemons; on the management host, the cluster file and hostcache file lists this information).

1. Log on to EGO as a cluster administrator:

   egosh user logon -u *user_name* **-x** *password*

   For example, type

   ```
   egosh user logon -u Admin -x Admin
   ```
2. Shut down all EGO services and daemons on all Windows hosts in the cluster:

   ```
   egoshutdown.bat
   ```

Shut down EGO on the Linux/UNIX cluster

Log on as `egoadmin` or `root` to any host in the cluster.

Note:

If the master is down, log on to a management host (if the master is down, there is no way to get the complete list of cluster services and daemons; on the management host, the cluster file and hostcache file lists this information).

1. Log on to EGO as a cluster administrator:

   egosh user logon -u *user_name* -x *password*

   For example, type

   egosh user logon -u Admin -x Admin
2. Shut down all EGO services and daemons on all Linux/UNIX hosts in the cluster:

   egoshutdown.sh

Restart EGO on the Windows cluster

You must be logged on to any host in the cluster as the local system user and have EGO administrative privileges.

Note:

If the master is down, log on to a management host (if the master is down, there is no way to get the complete list of cluster services and daemons; on the management host, the cluster file and hostcache file lists this information).

1. Restart EGO:

   ```
   egosh ego restart all
   ```

   EGO restarts on all hosts in the cluster.

Restart EGO on the Linux/UNIX cluster

Log on with root permissions on the local host. Assuming the master is up, logon to any host in the cluster.

Note:

If the master is down, log on to a management host (if the master is down, there is no way to get the complete list of cluster services and daemons; on the management host, the cluster file and hostcache file lists this information).

1. Restart EGO:

   ```
   egosh ego restart all
   ```

   EGO restarts on all hosts in the cluster (including Windows hosts if you have a mixed cluster). Services are not restarted with this command.

Start EGO in a mixed cluster

To start a Linux/UNIX host, use `root` (or the cluster administrator with root privileges). You cannot start a Linux/UNIX host from a Windows host.

To start a Windows host, use the Windows cluster administrator account. You cannot start a Windows host from a Linux/UNIX host.

1. Log on to a Windows host as the cluster administrator.
2. Start all Windows hosts in the cluster.

   egosh ego start all
3. Log on to a Linux/UNIX host as `root` (or the cluster administrator with root privileges).
4. Start all Linux/UNIX hosts in the cluster.

   egosh ego start all

   Note:

   You need to use rsh to remotely start a Linux/UNIX host.

Note:

On Windows 2008 with the default security policy settings and Symphony in advanced workload execution mode, if you installed with the Windows local system administrator account (Administrator), you must also use that account to start EGO. If the current logon user is not Administrator, you will need to input the password of the Administrator account every time you start EGO. To start the cluster, use the followingcommand: runas Administrator egosh ego start all

Shut down EGO in a mixed cluster

You need to be logged onto the master host as `root` (for Linux/UNIX master hosts) or as the cluster administrator with root privileges.

Shut down a mixed cluster as you would a normal cluster.

1. Log on to the cluster as `Admin`.

   egosh user logon -u Admin -x *Admin_password*
2. Run the shutdown script for your master host operating system type.

   Linux master host: **egoshutdown.sh**

   Windows master host: **egoshutdown.bat**

Important:

Using **egosh ego shutdown all** does not shut down the cluster properly.

Check hosts in a mixed cluster

1. View resources in the cluster.

   ```
   egosh resource view
   ```

   The operating system of the host can be seen in the "type" column.

About system services

A system service runs internal processes for Platform software. System services may have multiple concurrent service instances running on multiple hosts. All system services (except for derbydb) are automatically enabled by default at installation.

By default, management services that are installed with EGO include the following:

System Service	Process	Description
EGO service director	ServiceDirector	A system service that functions as a locating mechanism for other system services.
Platform Management Console	WEBGUI	A system service that runs the Platform Management Console.
EGO web service gateway	WebServiceGateway	A system service that provides a standards-based web services interface for web service clients (applications) to contact EGO.
EGO repository service	RS	A system service that manages package deployment. Allows deployment of a service without reliance on a shared file system.
PERF data purger	purger	A system service used by the reporting feature.
PERF loader controller	`plc`	A system service used by the reporting feature. Manages the data loaders that gather data from the system and writes the data into the database.
PERF derby database	`derbydb`	A small-footprint open source database that runs as a system service when first installed. This database is only appropriate for demo clusters. This service is only enabled if an environment variable is set prior to installation (Linux/UNIX) or during the Windows installation.
Session director	SD	The session director acts as a liaison between the client application and the session manager. There is one session director process per cluster and it runs on the master or a management host.
Session manager	SSM	Primary workload scheduler for an application.There is one session manager per application.

By default, system services are registered to the ManagementServices > EGOManagementServices consumer that uses the ManagementHosts resource group. You do not need to register any non-system services to consumers using the ManagementHosts resource group, although it is recommended that you do so if your service is a scheduler or controlling service instead of a regular application (management hosts are not expected to execute workload units for users, but are configured to run important services instead).

Each started system service requires one slot, assuming the system service is configured for one instance (default setting). By default, EGO allots twelve slots per management host to run system services, although only seven (or eight if `derbydb` is automatically enabled at installation) are actually used out-of-the-box by the installed management services listed above. For maximum performance in case of failover, you must retain seven or eight of the ten slots on each master candidate for use by the system services; the others can be used by non-system services that you might register that require a management host.

Slots assigned to run system services for consumers the ManagementServices consumer branch do not have to be on the master host (for example, any management host in the cluster can be the web server); the cluster determines where the system service instance runs at startup.

Registered system services do not require a host, only the system service instance that it runs.

Service profile

A service profile is an XML file (configured through the Platform Management Console) that contains the service definition for a service; when you update a system service, you are updating the system service profile containing the system service definition. Every service, whether it is installed by default (a system service) or added by an administrator, has a service profile. The XML file lets you configure properties for the service such as :

- The maximum and minimum number of service instances required by the service
- A service instance description, which describes how to start a service instance for this service
- Any resources required to run the service instances for this service
- Any dependencies this service has (such as another service being started)

Note:

We do not recommend changing the XML file for any of the system services (those installed by default) unless specifically told to do so. You may need to change ports.

Service name

If you plan on running a custom service as an EGO service, the service naming convention specifies that the maximum length of the service name is 40 characters and the first character can only be an alphabetic or hyphen. The remaining characters can be alphabetic, digits and hyphen.

Troubleshooting system services

If a system service has the state error or unknown, check the log files to troubleshoot. The logs for the EGO service controller, the load information manager, and the VEM kernel daemon are good places to start.

System services and EGO daemons

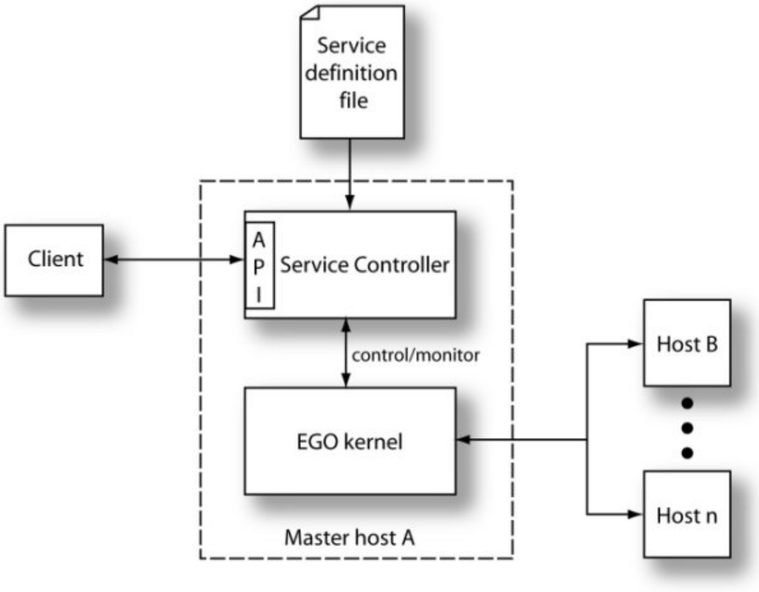
A system service is a self-contained, continuously running process that accepts one or more requests and returns one or more responses. Services may have multiple concurrent service instances running on multiple hosts. Most system services are automatically enabled by default at installation (derbydb is the exception, and must be manually enabled during installation).

About the service controller

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

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

The service controller also provides APIs to allow other tools to instantiate, control, and query services at runtime.



Service definition

The service controller reads configuration files for services that must be instantiated. The configuration files are XML documents (service profile) that contain the service definition (that is, parameters that define the type of resources the service instances need to run along with how to start and monitor the service instances). These files can be created either by the service controller administrator or by the API during runtime. There is one service definition file dedicated to each service.

Start-up sequence

Service controller start-up sequence:

1. One of the hosts within the cluster is elected as the master. The master host, in turn, starts the EGO kernel, which then starts the service controller on the same host.
2. The service controller opens a connection to EGO and registers as a recoverable client.
3. The service controller loads the service definition database containing the list of services that need to be instantiated.
4. The service controller recovers the latest state info from EGO and starts and stops services following the service definitions from the database. The service controller is ready for service.

About the service director

The service director is a basic system service that functions as a locating mechanism for other system services. The service director contains a stand-alone Domain Name Server (DNS), which is the authoritative name server for the EGO DNS sub-domain and responds to DNS queries for system services.

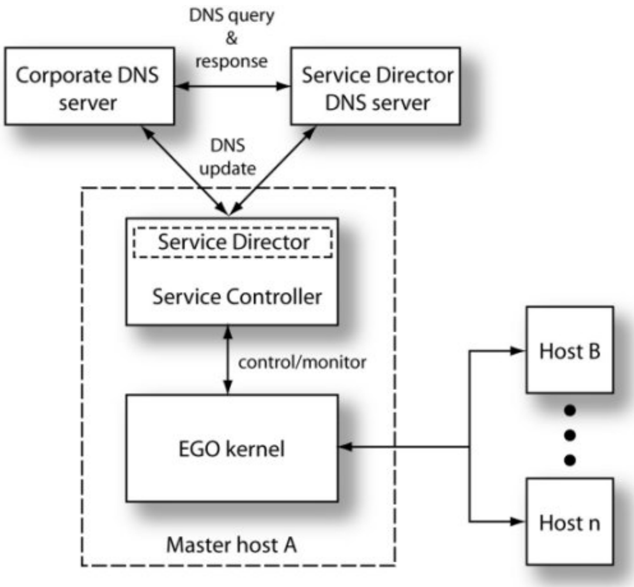
Restriction:

The service director is not supported by Simplified WEM.

The service director runs on the EGO master host and relies on the service controller to provide location information and state change notifications of service instances.

When a service instance enters the RUN state, the service director adds its location information into the service director DNS server. When a service instance transfers from the RUN state into other states, the service director deletes the location information from its DNS server.

When the locations of the service director DNS server or other system services are changed, the service director updates the corresponding resource record in the DNS database of the corporation DNS server or service director DNS server, respectively.



Service director DNS server

The service director DNS server is a stand-alone DNS server and has the responsibility to process DNS queries and maintain the mapping between server names and IP addresses. There is only one service director DNS server running in the EGO environment.

Service director start-up and recovery

The service director DNS server is essential to the operation of the service director. Consequently, the service director DNS server is configured with automatic startup, and therefore, the service controller starts the service director DNS server whenever it finds the DNS server is not running.

When the service controller restarts, it recovers the information of all the services from the EGO kernel. If the service controller finds the service director DNS server is still running, it recovers it.

During recovery, the service director performs the following steps:

1. Updates the service director DNS server location information

   Removes the old location information of the service director DNS server from the corporation DNS server, and then adds the new location information.
2. Updates services location information

   Removes the old location information of all current services, and then adds the current location information of services that have running instances.

Service instance location update

When a service instance is down and there is no other instance running on the same host, the service director deletes its location information from the service director DNS server. When a service instance is running, the service director adds its location information in the service director DNS server. The service director DNS server handles the duplicate location information.

About WEBGUI service and Web server

The WEBGUI service provides a high level view of running system services from the Platform Management Console. A cluster administrator can view detailed system service information and assess whether any actions are required for system services and service instances.

Service configuration can be done via the Platform Management Console. Configurations done through the Console are updated automatically in a service profile (XML file). The system service is also registered through the Platform Management Console.

The Web server is the host that runs the WEBGUI service (in effect, the Platform Management Console). Only one management host is elected as the Web server host.

About WebServiceGateway

The web service gateway (WebServiceGateway) service is a runtime component of EGO. The gateway provides a standards-based web services interface for web service clients to access EGO functionality. The web service client sends its request to the gateway via SOAP protocol. The gateway calls the EGO C APIs to perform the required operations on behalf of the web service client and returns the results.

Daemon file names

The following table outlines the EGO daemons and their associated log file names. Log files on Windows hosts have a `.txt` extension. Audit logs must be enabled first.

Daemon	**Log file name**
tomcat (WEBGUI)	`catalina.out`
datasourcetools (Database Configuration Tool)*	`datasourcetools.`*`hostname`*`.log`

Daemon	**Log file name**
egoconsumerresloader (Consumer Resource Data Loader)*	`egoconsumerresloader.`*`hostname`*`.log`
egodynamicresloader (Dynamic Metric Data Loader)*	`egodynamicresloader.`*`hostname`*`.log`
egoeventsloader (EGO Events Data Loader)*	`egoeventsloader.`*`hostname`*`.log`
egosc (EGO Service Controller)	`egoservice.audit.log`, `esc.log.`*`hostname`*
egostatisticresloader (Static Attribute Data Loader)*	`egostatisticresloader.`*`hostname`*`.log`
fam (File Access Manager)	`fam.`*`hostname`*`.log`
lim (Load Information Manager)	`lim.log.`*`hostname`*
named (Service Director)	`named.log`
pem (Process Execution Manager)	`pem.log.`*`hostname`*
pim (Process Information Manager)	`pim.log.`*`hostname`* (Linux only)
plc (Loader Controller)*	`plc.`*`hostname`*`.log`
purger (Data Purger)*	`purger.`*`hostname`*`.log`
rfa (Remote File Access)	`cli.`*`hostname`*`.log`

rs (Repository Service)	`rs.`*`hostname`*`.log,`
	`repositoryservice.audit.log`

vemkd (EGO Kernel Daemon)	`ego.audit.log`, `vemkd.log.`*`hostname`*
WSG (Web Service Gateway)	`wsg.log`
WSM (Platform Management Console/WEBGUI)	`wsm.log.`*`hostname`*

View a consumer’s system service

You must be a cluster administrator.

You can view the services that you (or the system) associated with a particular consumer. You can also view services for all consumers. Services are associated with consumers because they must exist somewhere in the consumer tree.

1. From System Services, click Monitor Services.

   The Services page displays.
2. In the consumer tree, select the consumer whose services you wish to view.

   Click the cluster name to select from a full list of existing services for all consumers or click a specific consumer name to select from a shorter list of services belonging to only one consumer.

   Note:

   A service can only be registered to a leaf. If your consumer has a sub-consumer, it is not a leaf.

System service states

System services have one of the following states:

State	Description
Defined	The service definition is loaded or created by API without syntax error. This state persists only if you defined a service to be started manually or if you have disabled it by API.
Init	The system service is being initialized. Dependencies are being checked. A transitional state before becoming `ALLOCATING`.
Allocating	Allocating resources to the system service. A transitional state before becoming `STARTED`.
Started	The system service is active and running. System service instances are running (the minimum number have been started).
Error	An error has been detected. Needs manual debugging.
Deallocating	The system service has been disabled by the system. Cleanup is occurring. A transitional state before becoming `DEFINED`.

About system service instances

System service instances are the result of a running system service. For example, system services that are stopped for any reason do not have system service instances.

A system service instance only runs one activity. You cannot control the activity directly, only the system service instance that contains it. For example, you can migrate an instance (and therefore the activity it contains), but you cannot migrate the activity directly.

You can have multiple system service instances running under a single system service; this means multiple activities can run simultaneously.

By default, system services have a configured number of system service instances to run on each slot (1 instance per slot).

System service instances require a host to run an activity. A system service instance runs using the resource group (for example, the ManagementHosts group or the ComputeHosts group) that is defined in the system services' definition files. Find the definition files here:

- Windows: `%EGO_TOP%\eservice\esc\conf\services`
- Linux/UNIX: `$EGO_TOP/eservice/esc/conf/services`

Note:

If you ran `egoconfig mghost`, then find the definition file in the *EGOshare* directory, where "*EGOshare*" is the shared directory containing important configuration files.

- Windows: `EGOshare\eservice\esc\conf\services`
- Linux/UNIX: `EGOshare/eservice/esc/conf/services`

System service instance states

All system service instances have one of the following states:

State	Description
Start	The system service instance is starting.
Run	The system service instance is running.
Finish	The system service instance has completed its task and has finished.
Unknown	The system service instance has not communicated and its current status is unknown. The execution host is unavailable for some reason. The activity is restarted on a new host if HostFailoverInterval is defined and reached.
Zombie	The system service running on an unreachable host has been killed.

A system service instance state is taken from the state of the activity that is running under it.

Determine the host address where a system service runs

You need to know the address of where the service director (DNS server) is running (contact IT for assistance, if required).

System services may not all run on the same management host. You can use `nslookup` to find the address of the host where a specific system service is running.

1. Using the CLI, type `nslookup`.
2. Type `server` and then enter the IP address of the service director (DNS server).

   The Default Server and Address returns. For example,

   ```
   > server 179.21.297.3 Default server: 179.21.297.3 Address: 179.21.297.3#53
   ```
3. Enter the name of the system service for which you want to find the host address. For example,

   ```
   > WEBGUI.egoServer: 179.21.297.3Address: 179.21.297.3#53Name:
   WEBGUI.egoAddress: 180.321.110.34
   ```

Shutting down custom system services gracefully

If you are running a custom system service as an EGO service, you can specify a script to gracefully shut down service instances. There is also a configurable timeout parameter during which the system waits for the target instance to exit. The system kills the target instance if it is still running after the timeout expires. This feature is implemented by the JobController and ControlWaitPeriod parameters.

About ControlWaitPeriod and JobController

When the service controller wants to kill (shut down) a service, it gives the service a "grace period" that is defined by <ego:ControlWaitPeriod>. The job control command will be started on the same host with the same initial environment as the container to be terminated. After the grace period has passed, if the instance container is still alive, SIGKILL is sent to terminate the container.

1. JobController:

   This script is prepared by the user to perform cleanup. If the job controller fails, EGO will forcibly terminate the service instance.
2. ControlWaitPeriod:

   Defines the grace period. If the instance container is still alive after the grace period has passed, PEM sends SIGKILL to terminate the container. Also, the job controller itself will be killed when SIGKILL is sent. The format of ControlWaitPeriod is PTXHYMZS, which means X hours, Y minutes and Z seconds. For example, PT10M0s means 10 min and PT60s means 60 sec. The range is 0~1hour; if the setting is out of this range, the service will not be loaded by egosc.

Configure ControlWaitPeriod and JobController

The ControlWaitPeriod and JobController parameters can be added to the Service Defintion file through the Platform Management Console. Follow the steps outlined in *Update a service* on page 30 to add and configure the parameters.

Verification

Check if your service can be stopped after a grace period instead of being killed immediately. The grace period might have 5+ seconds delay.

Also, you can check the egosc log under `$EGO_ESRVDIR/esc/log`. You should see:

```
2009-04-01 09:18:46.000 CST WARN [13769] do_containerStateChange(): on host
<bjg270-01>, the container <3> belongs to instance <1> of service <plc>
terminated, reason <Terminated by job controller>, status <1>
```

The message above means that the job controller has finished its clean up and terminated the service successfully.

Troubleshooting

Job Controller did not kill the service

Check the egosc log under $EGO_ESRVDIR/esc/log. You should see:

```
2009-04-01 09:17:17.000 CST WARN [13769] do_containerStateChange(): on host
<bjg270-01>, the container <9> belongs to instance <1> of service <test>
terminated, reason <Terminated by SIGKILL, job controller does not exist or
failed>, status <0>
```

The service cannot be loaded by EGOSC

1. Only ControlWaitPeriod was added to the Service Definition file.

   Check the egosc log for the following message:

   ```
   2009-04-01 11:49:31.000 CST ERROR [8946] validContainerSpec(): Conflict
   parameters, controlWaitPeriod is defined but JobController is not defined,
   refused2009-04-01 11:49:31.000 CST ERROR [8946] loadServiceDefinition():
   parse section ServiceDefinition failed

   2009-04-01 11:49:31.000 CST ERROR [8946] loadServiceDefinition():parse
   service definition file /opt/ego/eservice/esc/conf/services/test.xml
   failed

   2009-04-01 11:49:31.000 CST ERROR [8946] loadServices(): failed to load
   service definition from </opt/ego/eservice/esc/conf/services/test.xml>
   ```
2. ControlWaitPeriod in the Service Definition file is less than 0 or greater than 1 hour.

   Check the egosc log for the following message:

   ```
   2009-04-01 12:25:30.000 CST ERROR [10321] validContainerSpec(): Invalid
   controlWaitPeriod, refused

   2009-04-01 12:25:30.000 CST ERROR [10321] loadServiceDefinition(): parse
   section ServiceDefinition failed

   2009-04-01 12:25:30.000 CST ERROR [10321] loadServiceDefinition():parse
   service definition file /opt/ego/eservice/esc/conf/services/test.xml
   failed

   2009-04-01 12:25:30.000 CST ERROR [10321] loadServices(): failed to load
   service definition from </opt/ego/eservice/esc/conf/services/test.xml>
   ```

The service is terminated after 2 minutes

1. ControlWaitPeriod in the Service Definition file is configured as 0.

   If you define ControlWaitPeriod as PT0H0M0S, PT0M0S, or PT0S, EGO will set the value of ControlWaitPeriod to 2 minutes, and at the same time ControlWaitPeriod will be removed by egosc from the Service Definition file.
2. Only JobController was added to the Service Definition file.

   If you did not define ControlWaitPeriod but only JobController, the default value for ControlWaitPeriod is 2 minutes.

Update a service

You must be a cluster administrator. The service must be in the service state "DEFINED."

Services that are in other service states (such as "STARTED") have service profiles that can be viewed but not edited.

1. From System Services, click Configure Services.

   The Services page displays.
2. In your consumer tree, navigate to the location of the service you wish to update.

   You can only update services from a leaf location. If your consumer has any sub-consumers, then it is not a leaf.
3. Click Actions for the service you wish to edit, and select Open Profile from the drop-down list.

   The Service Profile dialog displays showing the configured parameters for the service.

   Note:

   You can only update services in a DEFINED state. Services in other states are read-only.
4. Click Table Preferences at the top of the dialog box, and check the boxes to add the corresponding information columns to the table.
5. To change any existing parameter values, click within the appropriate Value field and edit the value.
6. To add or remove a parameter, click Actions from the appropriate service section (for example, "sc:ServiceDefinition"), and then insert or remove the desired optional service parameter.
7. Click Save, confirm your changes, and then click Close.

Your service is now updated.

Register a new service

You must be a cluster administrator.

You can register a service if you need to add a new service or if you unregistered a service you need again.Registered services may need to be enabled afterwards if the service is set up to be manually started.

1. From System Services, click Configure Services.

   The Services page displays.
2. In your consumer tree, navigate to the location where you want to register your service.

   You can only register a service to a leaf location. If your consumer has any sub-consumers, then it is not a leaf.
3. From Global Actions, select Register a new service.

   The Service Profile displays showing the minimum required and preferred attributes for a new service along with preconfigured default values.
4. Click Table Preferences at the top of the dialog box and check the boxes to add the corresponding information columns to the table.
5. For main sections in the service profile (for example, "sc:ServiceDefinition"), click Actions and then insert any desired optional service parameters.
6. For each parameter, click within the Value field to add or change the value.

   Note:

   The actual number of slots requested with MaxInstances/ MaxInstancesPerSlot may be adjusted depending on the values of MaxInstancesPerSlot and MaxInstancesPerHost. For EGO service instances, the number of slots provided on each host is a fixed amount determined by MaxInstancesPerHost/MaxInstancesPerSlot. If the number of slots requested is not a multiple of the fixed amount, extra slots will be needed to fill the gap and thus be wasted. Take the following configuration as an example:

   ```
   <sc:MaxInstances>10</sc:MaxInstances>
   <sc:MaxInstancesPerSlot>1</sc:MaxInstancesPerSlot>
   <sc:MaxInstancesPerHost>6</sc:MaxInstancesPerHost>
   ```

   Ideally 10 slots (MaxInstances/MaxInstancesPerSlot) are needed. However, the integer value of MaxInstancesPerHost/ MaxInstancesPerSlot (fixed amount) is 6 and thus EGO will allocate 12 slots instead of 10, causing two slots to be wasted. To prevent slot wastage, ensure MaxInstances is a multiple of MaxInstancesPerHost/ MaxInstancesPerSlot.
7. Click Register and confirm that you want to modify your service.

Your service is registered. You may see the state change between DEFINED, INIT, ALLOCATING, and STARTED.

If your state remains DEFINED, you need to start your service.

CHAPTER

2

Host Management

Host states

Host state	Description
OK	Your host is functioning normally and has no trouble communicating with the system. OK hosts accept new workload. The host is participating in host scavenging. Priority may be set to normal or lowest priority.
Unavailable	One or more of the daemons running on the host have failed or are not communicating with the master host. An unavailable host does not accept new workload.
Closed	A host is closed when an administrator chooses to close or lock it. Administrators close a host to upgrade software or to troubleshoot hardware. Hosts can also be closed when host scavenging has been enabled and the host is no longer idle. If the host has already been allocated to a consumer to run Symphony work, the host continues to run Symphony workload until the consumer releases the resource. Once the host is released, it no longer accepts new work. Note that if a scavenged host is closed due to host thresholds being exceeded, the workload is either terminated immediately (if configured for fastrelease) or after a grace period.

Host properties

The console displays both static and dynamic host properties in a separate window. You can have up to four Host Properties windows open simultaneously to compare and monitor hosts.

Some of the most useful dynamic host properties are also displayed in chart form on the Charts tab.

Host properties and descriptions are listed below. Note the different host attributes associated with a property type:

- Built-in: The host property name and definition are preconfigured and built-in to EGO. Names and definitions cannot be changed.
- Reserved: The property name is built-in to EGO, and is reserved. A reserved host attribute must be configured explicitly using this name. Definitions are configurable.
- External: Names and definitions are completely configurable. They do not appear on the Host Properties page.

Property	Description	Attribute type
Host Name	Name of the host.	Built-in
Status	Current state of the host: OK, Unavailable, or Closed.	Built-in
Type (Host Type)	Type of host you have. For example, LINUX86.	Built-in
CPUs (Number of CPUs)	Number of ncpus you have specified for your host.	Built-in
CPU Util	Current CPU utilization of your host in %.	Built-in
Mem (Available Memory)	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.	Built-in
Swap (Available Swap)	Virtual memory (swap space) in MB currently available.	Built-in
Pg (Paging Rate)	Virtual memory paging rate in pages per second. This index is closely tied to the amount of available memory and the total size of the processes running on a host; if there is not enough memory to satisfy all processes, the paging rate is high.	Built-in
I/O (Disk I/O Rate)	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.	Built-in
Total Slots (Total Number of Slots)	Total number of slots for this host across ALL resource groups.	Built-in
Free Slots (Number of Free Slots)	Current number of available slots for this host across ALL resource groups.	Built-in
Host Close Comment	Comment associated with closing the host, if applicable.	Built-in

Property	**Description**	**Attribute type**
nprocs	Number of physical processors (if ncpus is defined as procs, then ncpus=nprocs).	Built-in
ncores	Number of cores per processor (if ncpus is defined as cores, then ncpus=nprocs x ncores).	Built-in
nthreads	Number of threads per core (if ncpus is defined as threads, then ncpus=nprocs x ncores x nthreads).	Built-in
numActivity	Number of activities running on the host..	Built-in
serverType	The host type. Valid values: dynamic (i.e., host dynamically joined the cluster) or static (i.e., host defined in ego.cluster.<clustername> file).	Built-in
15s Load (15-Second Load)	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.	Built-in
15m Load (15-Minute Load)	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.	Built-in
1m Load (1-Minute Load)	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.	Built-in
Model (Host Model)	Model of your host. For example, Intel_EM64T.	Built-in
Process Priority	OS process priority of cluster workloads (`normal` or `lowest`).	Built-in
Host Status Reason	Reason for the current host status, if applicable.	Built-in
CPU Factor	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.	Built-in
Max Mem	Maximum RAM available.	Built-in
Max Swap	Maximum swap space on your host.	Built-in
Temp (Available Temp)	Space available in MB on the file system that contains the temporary directory.	Built-in
Max Temp	Maximum space in `/tmp` (Linux/UNIX) or OS default temp directory (Windows).	Built-in
Disks	Number of local disks on your host.	Built-in

Property	Description	Attribute type
User Idle Time	Amount of time in minutes that a host has been idle. On a Linux/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.	Built-in
Users (Login Users)	Number of current users logged in to the system.	Built-in
Resource Attr	Resource attributes assigned to this host. For example, "mg" indicates the host is a management host; "scvg" indicates the host is scavenge-ready.	Built-in
CPU Speed	Speed of each individual CPU in MHz.	Built-in
Band Width	Maximum bandwidth requirement in Mbps.	Built-in
Scavenging Control	When host scavenging is enabled and the scavenging agent is controlling this host, value is on..	Reserved
User idle time threshold	User idle time threshold, in minutes; configurable external attribute.	Reserved
CPU idle time	Amount of time in minutes that a CPU has been idle.	Reserved
CPU idle time threshold	CPU idle time threshold, in minutes.	Reserved
Adjusted CPU utilization threshold	The user-defined threshold qpplied to the Adjusted CPU Utilization. The threshold is used as a trigger for host scavenging.	Reserved
Adjusted CPU utilization	The CPU utilization of the host without considering the CPU utilization of the specified exempt processes.	Reserved
Exempt Processes	Processes that are excluded from the calculation of Adjusted CPU Utilization.	Reserved
Close processes	Processes that, if running, will close the host..	Reserved
Host release mode	Release mode of the current scavengeable host Valid values: Fast release, Release with grace period.	Reserved

About load indices

Load indices measure the availability of dynamic resources on hosts. Dynamic resources are properties of a host that change as the load on that host changes, such as available memory and CPU utilization.

Load indices are not configurable; they simply reflect the current state of resources on your hosts given the current load they are managing. For example, as more workload units are assigned to and being run on host A, host A's CPU utilization load index increases. A load index may increase or decrease as the host's resources are put under more load. For example on host A, the total available memory decreases as the load increases.

How often are they measured?

Load indices are measured automatically at fixed time intervals. Each index is individually monitored and has its own update interval (from one of the shortest intervals at 15 seconds for status, to the longest interval at 120 seconds for available temporary space).

Why do I use them?

Viewing load indices for one host provides an excellent snapshot of how that host is performing at a specific moment. For troubleshooting purposes, you may want to track the load indices of one host over time (for example, an hour or a day).

Viewing load indices for all your hosts provides an overall snapshot of how your cluster is performing under its current load at a specific moment.

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

View host models and types

View detected host models and types

1. Run `egosh resource view [resource_name …]` to display information about host models that exist in the cluster.

View UNKNOWN and DEFAULT host models and types

Viewing UNKNOWN host type or model

1. Run `egosh resource view [resource_name …]`.

   A model or type UNKNOWN indicates the host is down or the lim on the host is down. You need to take immediate action.

Viewing DEFAULT host type or model

1. Run `egosh resource view [resource_name …]`. If model or type are displayed as DEFAULT and automatic host model and type detection is enabled, you can leave it as is or change it.

   If model is DEFAULT, EGO works correctly but the host has a CPU factor of 1, which may not make efficient use of the host model.

   If type is DEFAULT, there may be binary incompatibility. For example, there are two hosts, one is Solaris, the other is HP. If both hosts are set to type DEFAULT, it means jobs running on the Solaris host can be migrated to the HP host and vice versa.

Understanding how lim determines host models and types

The lim (load information manager) daemon/service automatically collects information about hosts in an EGO cluster, and accurately determines running host models and types. At most, 1024 model types can be manually defined in `ego.shared`:

- Windows: `%EGO_CONFDIR%\ego.shared`
- Linux/UNIX: `$EGO_CONFDIR/ego.shared`

If `ego.shared` is not fully defined with all known host models and types found in the EGO cluster, the lim attempts to match an unrecognized running host to one of the models and types that is defined.

The lim supports both exact matching of host models and types, and "fuzzy" matching (where an entered host model name or type is slightly different from what is defined in `ego.conf`).

How does "fuzzy" matching work?

The lim reads host models and types that have been manually configured in `ego.shared`. The format for entering host models and types in `ego.shared` is *model_bogomips_architecture* (for example, **x15_4604_OpterontmProcessor142**, **IA64_2793**, or **SUNWUltra510_360_sparc**). Names can be up to 64 characters long.

When the lim attempts to match running host model with what's entered in `ego.shared`, it first attempts an exact match, then proceeds to make a fuzzy match. Here is a summary on how the lim attempts to make matches, depending on given information:

Architecture name of running host	What the lim reports	Additional information about the lim process
Same as definition in `ego.shared` (exact match)	Reports the reference index of exact match	The lim detects an exact match between model and input architecture string

Architecture name of running host	What the lim reports	Additional information about the lim process
Similar to what is defined in `ego.shared` (fuzzy match)	Reports fuzzy match based on detection of 1or 2 fields in the input architecture string	• For input architecture strings with only one field: If the lim cannot detect an exact match for the input string, then the lim reports the best match; a "best match" is defined as a model field with the most characters shared by the input string
		• For input architecture strings with two fields: **1.** If the lim cannot detect an exact match, it attempts to find a best match by identifying the *model* field with the most characters that match the input string **2.** The lim then attempts to find the best match on the *bogomips* field
		• For architecture strings with three fields: **1.** If the lim cannot detect an exact match, it attempts to find a best match by identifying the *model* field with the most characters that match the input string **2.** After finding the best match for the model field, the lim attempts to find the best match on the *architecture* field **3.** The lim then attempts to find the closest match on the *bogomips* field, with wildcards supported (where the *bogomips* field is a wildcard)
Has an illegal name	Reports default host model	An illegal name is one that does not follow the permitted format for entering an architecture string where the first character of the string is not an English-language character.

Add a host type or model

The `ego.shared` file contains a list of host type and host model names for most operating systems. You can add to this list or customize the host type and host model names. A host type and host model name can be any alphanumeric string up to 39 characters long. The `ego.shared` file, which is accessed by the master host, is located in the EGO_CONFDIR directory.

1. Log on to the OS as the cluster administrator user.
2. Edit `ego.shared`:
    a) For a new host type, modify the `HostType` section:

```
Begin HostType
TYPENAME                     # Keyword
DEFAULT
IBMAIX564
LINUX86
LINUX64
NTX86
NTX64
NTIA64
SUNSOL
SOL732
SOL64
SGI658
SOLX86
HPPA11
HPUXIA64
MACOSX
End HostType
```

    b) For a new host model, modify the `HostModel` section:

    Add the new model and its CPU speed factor relative to other models.

```
Begin HostModel
MODELNAME  CPUFACTOR   ARCHITECTURE # keyword
# x86 (Solaris, Windows, Linux): approximate values, based on SpecBench results
# for Intel processors (Sparc/Win) and BogoMIPS results (Linux).
PC75             1.5   (i86pc_75  i586_75  x586_30)
PC90             1.7   (i86pc_90  i586_90  x586_34 x586_35 x586_36)
HP9K715          4.2   (HP9000715_100)
SunSparc          12.0         ()
CRAYJ90           18.0         ()
IBM350            18.0         ()
End HostModel
```

3. Save the changes to `ego.shared`.
4. Restart EGO on the master host so that the changes can take effect:

    egosh ego restart

Control hosts (Windows)

You can start, stop, and restart local, remote, and multiple Windows hosts. Find information on each of these options.

- Start hosts
  a) Start a local Windows host
  b) Start a remote Windows host
  c) Start multiple Windows hosts
- Shut down hosts
  a) Stop a local Windows host
  b) Stop a remote Windows host
  c) Stop multiple Windows hosts
- Enable automatic expiry of unavailable compute hosts
- Remove management hosts from the cluster
- Restart hosts
  a) Restart a local Windows host
  b) Restart a remote Windows host
  c) Restart multiple Windows hosts

Start hosts

Starting hosts brings them into the cluster where they become usable resources.

Start a local Windows host

Log on as egoadmin.

To start a local host, perform the following steps:

1. Start an interactive command console.
2. Start EGO on your local host.

   ```
   egosh ego start
   ```

Note:

On Windows 2008 with the default security policy settings and Symphony in advanced workload execution mode, if you installed with the Windows local system administrator account (Administrator), you must also use that account to start EGO. If the current logon user is not Administrator, you will need to input the password of the Administrator account every time you start EGO. To start, use the followingcommand: runas Administrator egosh ego start

Start a remote Windows host

Log on as egoadmin. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Start EGO on your remote host:

   egosh ego start *host_name*

   Replace *host_name* with the name of your remote host.

Note:

On Windows 2008 with the default security policy settings and Symphony in advanced workload execution mode, if you installed with the Windows local system administrator account (Administrator), you must also use that account to start EGO. If the current logon user is not Administrator, you will need to input the password of the Administrator account every time you start EGO. To start, use the followingcommand: runas Administrator egosh ego start host_name

---

Start multiple Windows hosts

Log on as egoadmin. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Start EGO on multiple hosts:

   egosh ego start *host_name host_name* ...

   Replace *host_name* with the names of your remote hosts. Separate host names with a space.

Note:

On Windows 2008 with the default security policy settings and Symphony in advanced workload execution mode, if you installed with the Windows local system administrator account (Administrator), you must also use that account to start EGO. If the current logon user is not Administrator, you will need to input the password of the Administrator account every time you start EGO. To start, use the followingcommand: runas Administrator egosh ego start host_name...

Shut down hosts

Shutting down a host immediately changes the host state to `Unavailable`. Running workload is automatically restarted on another host. If you need to perform maintenance, you can choose to close a host instead of shutting it down.

Removing a management host requires following a separate procedure.

Stop a local Windows host

Log on as egoadmin.

1. Stop EGO on your local host:

   egosh ego shutdown

Stop a remote Windows host

Log on as egoadmin. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Stop EGO on your remote host:

   egosh ego shutdown *host_name*

   Replace *host_name* with the name of your remote host.

Stop multiple Windows hosts

Log on as egoadmin. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Stop EGO on more than one host:

   egosh ego shutdown *host_name host_name* ...

   Replace *host_name* with the names of your remote hosts. Separate host names with a space.

Enable automatic expiry of unavailable compute hosts

By default, all hosts that join the cluster remain in the cluster, even if they become unusable.

Optionally, after a period of time in the `Unavailable` state, a compute host can expire from the cluster —it no longer appears in the Console, or in `egosh resource list` or `resource view` output, and it is not affected by `egosh start all` or `restart all` commands.

Host expiry is not irreversible. If you restart an expired host (for example, run `egosh ego start` *host_name* or restart the host while automatic system startup is configured), it can rejoin the cluster in the same way that a newly installed host joins the cluster.

If you want to remove a management host from the cluster, a different procedure is required.

To configure the host expiry feature, take the following steps.

1. Logon as `egoadmin` and edit `ego.conf`.
2. Specify a time-out period for the parameter `EGO_DYNAMIC_HOST_TIMEOUT`.

   The default time period is in hours. Use an M after the time value to represent minutes (for time periods of 10 minutes or more).

   For example:

   - `EGO_DYNAMIC_HOST_TIMEOUT=48` means the unavailable host is removed after 48 hours.
   - `EGO_DYNAMIC_HOST_TIMEOUT=75M` means the host is removed after an hour and fifteen minutes.
   - `EGO_DYNAMIC_HOST_TIMEOUT=2M` means the host is removed after 10 minutes; a value of 2 minutes is below the allowable 10 minute minimum.
3. Restart the master host for the change to take effect.

Remove management hosts from the cluster

You must have previously enabled automatic expiry of unavailable compute hosts.

Hosts that have been added to a cluster using the `egoconfig mghost cmd` and are designated as management hosts need to be physically deleted from `ego.cluster.`*`cluster_name`* if you want to remove them from the cluster. This requirement applies to current management hosts.

1. Shut down the host.
2. If you have configured automatic startup during your cluster setup, then run `egoremoverc.sh`.

   Doing this prevents automatic startup when the host reboots, which keeps the host from being re-added dynamically to the cluster.
3. If the host is a master candidate, run `egoconfig masterlist` to remove the host from the failover order.
4. Run `egoconfig unsetmghost` to remove the host from the management host group.

   Running this command removes the host entry from ego.cluster.*cluster_name*.
5. Run the installer (`MSI` file) on the host you wish to remove from the cluster, and uninstall the EGO package.
6. Restart the master host to change it from a management host to a compute host.

   Because the host is shut down, and daemons are no longer running, the host switches to an unavailable state. Now that you've got a compute host in an unavailable state, complete the steps for enable automatic expiry of unavailable compute hosts to remove the host from the cluster.

Restart hosts

You may want to restart a host if it has become unavailable to the cluster. An unavailable host may have problems with memory or unnecessary applications that can be fixed by restarting it.

Restart a local Windows host

Log on as egoadmin.

1. Restart EGO on your local host:

   egosh ego restart

Restart a remote Windows host

Log on as egoadmin. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Restart EGO on your remote host:

   egosh ego restart *host_name*

   Replace *host_name* with the name of your remote host.

Restart multiple Windows hosts

Log on as egoadmin. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Restart EGO on multiple hosts:

   egosh ego restart *host_name* *host_name* ...

   Replace *host_name* with the names of your remote hosts. Separate host names with a space.

Control hosts (Linux/UNIX)

You can start, shut down, and restart local, remote, or multiple Linux/UNIX hosts. Find information on each of these options.

- Start hosts
  a) Start a local Linux/UNIX host
  b) Start a remote Linux/UNIX host
  c) Start multiple Linux/UNIX hosts
- Shut down hosts
  a) Stop a local Linux/UNIX host
  b) Stop a remote Linux/UNIX host
  c) Stop multiple Linux/UNIX hosts
- Enable automatic expiry of unavailable compute hosts
- Remove management hosts from the cluster
- Restart hosts
  a) Restart a local Linux/UNIX host
  b) Restart a remote Linux/UNIX host
  c) Restart multiple Linux/UNIX hosts

Start hosts

Log on with root permissions and rsh on all hosts in the cluster. To start a 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.

Important:

By default, only root can start, stop, or restart the cluster. Optionally, you can grant root privileges to egoadmin, the cluster administrator account.

Starting hosts brings them into the cluster where they become usable resources.

Start a local Linux/UNIX host

Log on with root permissions.

To start a local host, take the following steps.

1. Start EGO on your local host:

   egosh ego start

Start a remote Linux/UNIX host

Log on with root permissions and rsh on the local host and the remote host. If the master is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Start EGO on a remote host:

   egosh ego start *host_name*

   Replace *host_name* with the name of your remote host.

Start multiple Linux/UNIX hosts

Log on with root permissions and rsh on the local host and each remote host. If the master is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Start EGO on multiple hosts:

   `egosh ego start` ***host_name host_name ...***

   Replace *host_name* with the names of your remote hosts. Separate host names with a space.

Shut down hosts

Shutting down a host immediately changes the host state to `Unavailable`. Running workload is automatically restarted on another host. If you need to perform maintenance, you can choose to close a host instead of shutting it down.

Stop a local Linux/UNIX host

Log on with root permissions.

1. Stop EGO on your local host:

   egosh ego shutdown

Stop a remote Linux/UNIX host

Log on with root permissions on the local host. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Stop EGO on a remote host:

   `egosh ego shutdown` ***host_name***

   Replace *host_name* with the name of your remote host.

Stop multiple Linux/UNIX hosts

Log on with root permissions on the local host. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Stop EGO on multiple hosts:

   `egosh ego shutdown` ***host_name host_name ...***

   Replace *host_name* with the names of your remote hosts. Separate host names with a space.

Enable automatic expiry of unavailable compute hosts

By default, all hosts that join the cluster remain in the cluster, even if they become unusable.

Optionally, after a period of time in the `Unavailable` state, a compute host can expire from the cluster —it no longer appears in the Console, or in `egosh resource list` or `resource view` output, and it is not affected by `egosh start all` or `restart all` commands.

Host expiry is not irreversible. If you restart an expired host (for example, run `egosh ego start` *host_name* or restart the expired host while automatic system startup is configured), it can rejoin the cluster in the same way that a newly installed host joins the cluster.

If you want to remove a management host from the cluster, a different procedure is required.

To configure the host expiry feature, take the following steps.

1. Logon as `egoadmin` and edit `ego.conf`.
2. Add the parameter `EGO_DYNAMIC_HOST_TIMEOUT` and specify an expiry period.

   The default time period is in hours. Use an M after the time value to represent minutes (for time periods of 10 minutes or more).

   For example:

   - `EGO_DYNAMIC_HOST_TIMEOUT=48` means the unavailable host is removed after 48 hours.
   - `EGO_DYNAMIC_HOST_TIMEOUT=75M` means the host is removed after an hour and fifteen minutes.
   - `EGO_DYNAMIC_HOST_TIMEOUT=2M` means the host is removed after 10 minutes; a value of 2 minutes is below the allowable 10 minute minimum.
3. Restart the master host for the change to take effect.

Remove management hosts from the cluster

You must have previously enabled automatic expiry of unavailable compute hosts.

Hosts that have been added to a cluster using the command `egoconfig mghost`, and are designated as management hosts, need to be physically deleted from `ego.cluster.`*`cluster_name`* if you want to remove them from the cluster. This requirement applies to current management hosts.

1. Shut down the host.
2. If you have configured automatic startup during your cluster setup, then run the command `egoremoverc.sh` as root.

   Doing this prevents automatic startup when the host reboots, and keeps the host from being re-added dynamically to the cluster.
3. If the host is a master candidate, run the command `egoconfig masterlist` to remove the host from the failover order.
4. Run `egoconfig unsetmghost` to remove the host from the management host group.

   Running this command removes the host entry from `ego.cluster.`*`cluster_name`*.
5. Optional. If you want to completely remove the host from the cluster, you can shutdown all the applicable services and daemons at this point, and then remove the directory where the Linux/UNIX installation is located.
6. Restart the master host to change it from a management host to a compute host.

   Because the host is shut down, and daemons are no longer running, the host switches to an unavailable state. Now that you've got a compute host in an unavailable state, complete the steps for Enable automatic expiry of unavailable compute hosts.

Restart hosts

You may want to restart the EGO daemons on a host if it is in an `Unavailable` state. An unavailable host may have problems with memory or unnecessary applications that can be fixed by restarting it.

Restart a local Linux/UNIX host

Log on with root permissions.

1. Restart EGO on your local host:

```
egosh ego restart
```

Restart a remote Linux/UNIX host

Log on with root permissions. If the master host is up, log onto any host in the cluster. If the master is down, log onto a management host.

1. Restart EGO on a remote host:

   egosh ego restart *host_name*

   Replace *host_name* with the name of your remote host.

Restart multiple Linux/UNIX hosts

Log on with root permissions. If the master host is up, log onto any host in the cluster; if the master is down, log onto a management host.

1. Restart EGO on multiple hosts:

   egosh ego restart *host_name host_name* ...

   Replace *host_name* with the names of your remote hosts. Separate host names with a space.

Add a compute host to the cluster (dynamically)

Install the Platform package for your host. The same master and port number as the rest of the cluster's hosts must be configured.

Compute hosts can be added to the cluster and included in a dynamic resource group automatically so they can be used immediately. This does not apply to management hosts. For information about adding management hosts, refer to the installation documentation.

1. Start Symphony on the host.

   The host contacts the LIM and joins the cluster.
2. If you have a dynamic resource group in your cluster and the host matches the resource requirement of it, the host is added to the resoure group automatically.

To specify the resources of a dynamically added host, specify them in the EGO_LOCAL_RESOURCES parameter in `ego.conf`.

Remove a compute host from the cluster

When a host has been added to the cluster dynamically, you can remove it from the cluster again if needed.

1. Close the host you want to remove.

   From Hosts (List), locate the host you want to close and select Actions > Close.
2. Shut down EGO on the host.

   ```
   egosh ego shutdown
   ```

   The host becomes unavailable after about 2 minutes. If the host is being used by any application, the SSM releases the resource to EGO and asks for a replacement. Any tasks running on that host are re-allocated on a different host. EGO does not allocate work to the closed host.

You can physically remove your machine from the cluster (or its connection to the network).

Transferring files from one host to another

Using the command line interface, you can transfer files between hosts. This is useful when installing or upgrading packages. Commands associated with data and file transfer include rfa list, rfa get, rfa put, and rfa remove.

- List host files
- Copy a file from another host
- Copy a file to another host
- Remove a file from a host

List host files

You must have appropriate privileges on each of the hosts involved. Directory path names must be absolute. You may wish to first request a resource allocation/slot from EGO before running the commands (if you do not, an allocation is automatically requested on your behalf).

Run the command `rfa list` to list file(s) from a local or remote directory.

1. From the command line, run the `rfa list` command.

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

   - **-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 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.

     Note:

     If you have a sequence of commands, there is no need log on or provide credentials each time. Note that credentials, which are locally stored, expire after a certain length of time.
   - **-u** *user_name*: (Optional.) Fully-qualified user name of the execution account to register the password for.
   - **-x** *password*: (Optional.) Password to register for the execution user account.

Note:

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 user_name and password.

Copy a file from another host

You must have appropriate privileges on each of the hosts involved in the file transfer. Directory path names must be absolute. You may wish to first request a resource allocation/slot from EGO before running the commands (if you do not, an allocation is automatically requested on your behalf).

Run the command `rfa get` to copy a file from a specified host to a local destination.

1. From the command line, run the `rfa get` command.

`rfa get -t` *host_name* `-d` *local_file* `| -s` *remote_file* `-p` *consumer* `[-c` *credential* `| -u` *user_name* `-x` *password*`]`

- **-t** *host_name*: Name of host you want to copy a file from.
- **-d** *local_file*: Name of the local file, including its directory location, you want to copy to. Directory path must be absolute
- **-s** *remote_file*: Name of the remote file, including its directory location, you want to copy. Directory path must be absolute.
- **-p** *consumer*: 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.

Note:

If you have a sequence of commands, there is no need log on or provide credentials each time. Note that credentials, which are locally stored, expire after a certain length of time.

- **-u** *user_name*: (Optional.) Fully-qualified user name of the execution account to register the password for.
- **-x** *password*: (Optional.) Password to register for the execution user account.

Note:

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 user_name and password.

Copy a file to another host

You must have appropriate privileges on each of the hosts involved. Directory path names must be absolute. You may wish to first request a resource allocation/slot from EGO before running the commands (if you do not, an allocation is automatically requested on your behalf).

Run the command `rfa put` to copy a file to a specified host from a local location.

1. From the command line, run the `rfa put` command.

   `rfa put -t` *host_name* `-d` *local_file* `| -s` *remote_file* `-p` *consumer* `[-c` *credential* `| -u` *user_name* `-x` *password*`]`

   - **-t** *host_name*: Name of host you want to copy a file to.
   - **-d** *local_file*: Name of the local file, including its directory location, you want to copy. Directory path must be absolute.
   - **-s** *remote_file*: Name of the remote file, including its directory location, you want to copy to. Directory path must be absolute.
   - **-p** *consumer*: 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.

   Note:

   If you have a sequence of commands, there is no need log on or provide credentials each time. Note that credentials, which are locally stored, expire after a certain length of time.

- **-u** *user_name*: (Optional.) Fully-qualified user name of the execution account to register the password for.
- **-x** *password*: (Optional.) Password to register for the execution user account.

Note:

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 user_name and password.

Remove a file from a host

You must have appropriate privileges on each of the hosts involved. Directory path names must be absolute. You may wish to first request a resource allocation/slot from EGO before running the commands (if you do not, an allocation is automatically requested on your behalf).

Run the command `rfa remove` to remove a file from a specified host.

1. From the command line, run the `rfa remove` command.

   `rfa remove -t` *host_name* `-d` *local_file* `| -s` *remote_file* `-p` *consumer* `[-c` *credential* `| -u` *user_name* `-x` *password*`]`

   - **-t** *host_name*: Name of host you want to remove a file from.
   - **-d** *local_file*: Name of the local file, including its directory location, you are removing. Directory path must be absolute.
   - **-s** *remote_file*: Name of the remote file, including its directory location, you are removing. Directory path must be absolute.
   - **-p** *consumer*: 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.

     Note:

     If you have a sequence of commands, there is no need log on or provide credentials each time. Note that credentials, which are locally stored, expire after a certain length of time.
   - **-u** *user_name*: (Optional.) Fully-qualified user name of the execution account to register the password for.
   - **-x** *password*: (Optional.) Password to register for the execution user account.

Note:

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 user_name and password.

CHAPTER

3

Controlling Symphony

Start Symphony cluster

You need to be logged onto the master host as `root` (for Linux/UNIX master hosts) or as the cluster administrator with root privileges.

1. Log on to the cluster as `Admin`.

   egosh user logon -u Admin -x *Admin_password*
2. Run the following commands for your master host operating system type.
   a) Start EGO

      ```
      egosh ego start all
      ```
   b) Enable application(s) if required

      `soamcontrol app enable` *application_name*

Note:

On Windows 2008 with the default security policy settings and Symphony in advanced workload execution mode, if you installed with the Windows local system administrator account (Administrator), you must also use that account to start EGO. If the current logon user is not Administrator, you will need to input the password of the Administrator account every time you start EGO. To start the cluster, use the followingcommand: runas Administrator egosh ego start all

---

Shut down Symphony cluster

You need to be logged onto the master host as `root` (for Linux/UNIX master hosts) or as the cluster administrator with root privileges.

1. Log on to the cluster as `Admin`.

   egosh user logon -u Admin -x *Admin_password*
2. Run the following commands for your master host operating system type.
   a) Disable all applications

      ```
      soamcontrol app disable all
      ```
   b) Stop all services

      ```
      egosh service stop all
      ```
3. Log on as OS user and shutdown the cluster

   `egosh ego shutdown` all

Licensing Platform Symphony products and features

Symphony supports two base license keys:

- sym_base
- sym_cluster_edition_base

Either one of these licenses is required for Symphony to function. If both licenses are present, the sym_base license takes effect. Symphony licenses are stored in a text file named `license.dat`, by default.

License key format

FEATURE [*feature_name*] lsf_ld [version] [expiryDate] 0 ADAD246AB126D5936EA9 HOSTID=[id] NOTICE="MAXCORE(#core)"

- Feature name: for example, sym_base
- Version: Symphony version, for example, 5.1
- ExpiryDate: Expiry date of the license.
- NOTICE: Defines the number of core licenses.

For example, a sym_base license key for 300 cores would look like this:

FEATURE sym_base lsf_ld 5.100 10-Dec-3010 0 ADAD246AB126D5936EA9 HOSTID=0080c84b0430 NOTICE="MAXCORE(300)"

sym_base license

A sym_base license allows a cluster to run an unlimited number of enabled applications concurrently and either an unlimited number of cores or the number of cores defined in NOTICE="MAXCORE(#core)".

The cores are reported by the OS, including cores on both physical and virtual hosts.

This license can be used with all add-on products. (Add-on product licenses are required.)

sym_cluster_edition_base

A sym_cluster_edition license allows a cluster to run up to 10 enabled applications concurrently and a maximum of 1000 cores or the number of cores defined in NOTICE="MAXCORE(#core)". The maximum number of cores that can be defined in the NOTICE is 1000.

The cores are reported by the OS, including cores on both physical and virtual hosts.

You can register more than 10 applications, but only 10 applications can be enabled at the same time.

Additional hosts that cause the cluster to exceed the core limit are not usable, but are still visible.

This license can be used with the following add-on products: (Add-on product licenses are required.)

- Data Affinity
- Dynamic Service

Licensing behavior

- The master LIM does not communicate with a license server, but instead verifies the license locally.

- A host is unlicensed if there is not enough core licenses for part or all of its cores.
- An unlicensed host receives a closed status, which cannot be opened.
- If a cluster initially has an unlicensed host and a licensed host is removed, the original unlicensed host will become licensed and opened automatically.

Licensing add-on products

Feasture licenses are required for the following add-on products:

Add-on product	license key
desktop scavenging	sym_desktop_scavenging
server scavenging	sym_server_scavenging
Adaptive Cluster	adaptive_cluster_for_symphony
data-aware scheduling	symphony_data_aware_scheduling
dynamic service	symphony_dynamic_service
MultiCluster	ego_base

The NOTICE="MAXCORE(#core)" parameter, which specifies the maximum number of core licenses for a feature, is defined in the following add-on product licenses:

- sym_desktop_scavenging
- sym_server_scavenging
- adaptive_cluster_for_symphony

The total number of cores supported in a cluster is the sum of:

MAXCORE(#core) for the three add-on product licenses and the MAXCORE(#core) of sym_base or sym_cluster_edition_base. (Note: If both licenses are present, the sym_base license takes effect.)

For example:

`1200 core cluster`

- sym_desktop_scavenging NOTICE="MAXCORE(#500)"
- sym_server_scavenging NOTICE="MAXCORE(#100)"
- adaptive_cluster_for_symphony NOTICE="MAXCORE(#100)"
- sym_base NOTICE="MAXCORE(#500)"
- sym_cluster_edition_base NOTICE="MAXCORE(#800)"

`800 core cluster`

- sym_desktop_scavenging NOTICE="MAXCORE(#100)"
- sym_server_scavenging NOTICE="MAXCORE(#100)"
- adaptive_cluster_for_symphony NOTICE="MAXCORE(#100)"
- sym_cluster_edition_base NOTICE="MAXCORE(#500)"

To view Symphony add-on product licensing using the PMC, select Cluster > Summary > License; refer to the PMC help for more information.

Updating or editing a license file

When a new license file is inserted into the cluster or an exisiting license file is updated, the following steps must be performed for the license change to take effect.

1. Restart LIM on the master and all master candidate hosts.
2. Stop and restart the SD service.
3. If the license file contains changes to the adaptive_cluster_for_symphony license key, stop and restart the IS service.
4. Disable and enable all applications.

How to view licensing information using the CLI

To view information about the number of cores licensed and which add-on products are licensed, use the `egosh license info` command; refer to the Symphony Reference guide for more details.

How to view unlicensed hosts using the CLI

You can use the `egosh resource view` command to check the host status reason for closed hosts. If a host is closed because it is unlicensed, it is stated as the reason. Refer to the Symphony Reference guide for more details.

4

Monitoring Resource Allocation

About resource allocation monitoring

Resource allocation monitoring, available only in the Platform Management Console, allows you to view your current owned, guaranteed, and shared resource allocation usage.

Resource allocation information

Data is current

The information shown is the most current data.

For historical analysis of resource allocation patterns over time, use the Reporting feature.

Host data is aggregated

The information shown combines data from all hosts. If a consumer is configured to use multiple resource groups, you cannot know how many of the consumer's slots are from each resource group.

Bar chart with a logarithmic scale

Allocation and demand is measured by number of slots. The information is displayed using a bar chart with a logarithmic scale that can indicate both small and large values. The scale reaches to 100,000 slots, regardless of cluster size or workload volume (the scale can indicate when a value exceeds 100,000 but the values can no longer be compared graphically).

Because the scale is not proportional, you should always roll over the chart and check the numeric values to best judge the allocation to each consumer.

Data filtering options

Depending on your focus in the tree, you can view resource allocation information for the entire cluster, one branch, or a single consumer.

List sorting options

By default, the list shows consumers organized hierarchically. You can sort by other criteria, such as Current Demand. The default then is to sort in ascending order, but you can reverse the sorting order by repeating the selection of the sort criteria.

Data available per consumer

For each consumer, you can view the following:

Allocation and Shortfall	Allocation is the number of slots allocated to the consumer. Shortfall occurs when a consumer's currently allocated slots (in non-management resource groups) are fewer than the number of guaranteed slots and there is an unsatisfied demand on those resource groups. This could occur if you have too few slots available in the hosts belonging to your resource group and that resource group has too many consumers drawing from it.
Owned, Shared, and Borrowed	The allocation is subdivided into the number of owned slots allocated to the consumer, the number of slots allocated to the consumer from the shared pool, and the number of slots borrowed by the consumer.

	When you roll over different colored sections of the bar chart, you see the numeric values for owned, shared, and borrowed.
Total Demand and Unsatisfied Demand	Total demand is the total number of slots requested by the consumer. Some or all of these slots may be allocated by EGO. Unsatisfied demand is the number of additional slots needed by the consumer to satisfy all demand. It is the difference between the number of slots requested and the number of slots allocated. When you roll over different colored sections of the bar chart, you see the numeric value for demand and unsatisfied demand.
Guaranteed	Guaranteed slots are the minimum number of slots to be allocated to the consumer if the consumer has demand. The guaranteed number of slots depends on the resource plan. The calculation adds the number of owned slots to the number of shared slots that the consumer should get if all consumers had maximum demand.
Cluster Size	Cluster size is the total number of slots in the cluster. This is shown for comparison purposes on each bar chart.

Relative allocation data

Optionally, you can see the current allocation in a pie chart, which shows the proportional distribution of slots at a glance. The legend identifies the 5 consumers that have the greatest allocation. Roll over the chart to see details for each consumer. This chart does not compare the current allocation to the resource plan.

For example, at the cluster level, you can see how many slots are allocated to each consumer, and how many slots are unallocated.

You can view relative allocation at the cluster level or at any branch of the consumer tree.

View resource allocation information

The bar chart shows detailed information for each consumer using a logarithmic scale that allows you to see details of each consumer.

Use sorting to help you compare consumers. For example, sort by Outstanding Demand to compare consumers most in need of additional resources with consumers least in need (top and bottom of list).

1. In the Console, navigate to Resources > Monitor Resource Allocation.
2. In the tree, select the focus (entire cluster or tree branch).

   A list of consumers appears, showing resource allocation information for each.

Compare relative resource allocation

The relative allocation chart shows consumers in a pie chart, to quickly judge the proportional allocation of resources among consumers.

1. In the Console, navigate to Resources > Monitor Resource Allocation.
2. In the tree, select the focus (entire cluster or tree branch).

   A list of consumers appears.
3. Click Show Relative Allocation.

   A pie chart shows the relative allocation of hosts to the consumers. The legend identifies up to five consumers.

View resource allocation shortfall

Shortfall occurs when a consumer’s currently allocated slots (in non-management resource groups) are fewer than the number of guaranteed slots and there is an unsatisfied demand on those resource groups. This could occur if you have too few slots available in the hosts belonging to your resource group and that resource group has too many consumers drawing from it. You should add more member hosts to an under-allocated resource group or assign more resource groups to the consumers involved.

You can quickly identify if this condition exists in either the cluster or in one branch of the tree.

1. In the Console, navigate to Resources > Monitor Resource Allocation.
2. In the tree, select the focus (entire cluster or tree branch).

   A list of consumers displays.
3. Sort by Under Allocation and repeat the sort selection to list consumers in descending order.

   If any consumer is under allocated, the Current Allocation is highlighted and the details of the under allocation are displayed.

View resource allocation details for a consumer

You may view details such as the number of slots borrowed by a consumer.

1. In the Console, navigate to Resources > Monitor Resource Allocation.
2. In the tree, select the focus (entire cluster or tree branch).

   A list of consumers displays.
3. For each consumer, roll over different colored sections of the bar chart to see numeric values for owned, shared, and borrowed.

CHAPTER

5

Reporting

About reports

About producing reports

The reports stored in the system do not include actual data. Instead, the reports define what data to extract from the system, and how to display it graphically.

Reports need to be produced before you can see the data. When you produce a report, you query the database and extract specific data. The amount of system overhead depends on how much data is in the report.

Reports have configurable parameters so you can modify the report and get exactly the data that you want.

About exporting reports

Data expires from the database periodically, so producing a report at a later date may return different data, or return no output at all. After you produce a report, you can keep your results by exporting the report data as comma-separated values in a CSV file. In this way you can preserve your data outside the system and integrate it with external programs, such as a spreadsheet. You can also keep your graphical results by using your browser to save the report results as an image.

Standard reports

For your convenience, Platform provides several standard reports for you to use. These reports allow you to keep track of some useful statistics in your cluster.

Standard reports are based on raw data stored in the relational database, and do not perform any data aggregation or calculations.

The following is a list of the standard reports that are included with the reporting feature. For further details on a report, view the full description of a standard report using the Console.

Note:

Depending on what you have installed, not all reports are available.

Name	Description	Category
Active Session or Task Statistics	Active tasks (running and pending) or sessions (open and suspended) in a cluster.	Symphony
Cluster Availability - EGO	EGO host availability in a cluster.	EGO
Cluster Slot Utilization - EGO	Percentage of total slots used in the cluster, averaged hourly.	EGO
Finished Session Hourly Statistics	Closed and aborted sessions in your cluster.	Symphony
Finished Task Hourly Statistics	Done, error, and canceled tasks in your cluster.	Symphony
Host Resource Usage	Resource usage trends for selected hosts.	EGO
Resource Allocation v. Resource Plan	Actual resource allocation compared to resource plan and unsatisfied resource demand for the selected consumer.	EGO
Session Hourly Throughput	Submitted and finished sessions in a cluster, aggregated hourly.	Symphony

Name	Description	Category
Top 5 Sessions Ordered by Attribute	List of sessions ordered by the selected attribute.	Symphony

Custom reports

You can create and use custom reports if the standard reports are insufficient for your needs.

While standard reports are provided for your use by Platform, custom reports are reports you create as needed to satisfy specific reporting needs at your site.

Custom reports let you define combinations of data that are not available in the standard reports. Custom report output is always displayed in tabular format.

Create custom reports

The easiest way to create a custom report is to copy an existing report, then customize the SQL query string as desired. To customize the SQL query string, you may need to refer to the data schema, which describes the organization of information in the relational database. The data schema for each standard report is available in the Console by opening the report and clicking Help.

Even if you cannot edit SQL, saving a report as a custom report lets you re-use the report data without having to re-input the parameters in the standard report.

- If the time period is fixed, you get the same data every time you produce the report, but the report will be empty when the data expires from the database.
- If the time period is relative, you can get data for a different time period each time you produce the report.

You can also define custom reports from a blank template and input the SQL query string directly.

When you create custom reports, you can enter a category and use it to group the reports any way you want.

Delete custom reports

Unlike standard reports, custom reports can be deleted. You might prefer to rename old reports (by modifying them) instead of deleting them.

Reports directory

The reporting feature resides in the `perf` directory, which is a subdirectory of the top-level EGO directory.

If you ran `egoconfig mghost` to configure management hosts, the top-level EGO directory refers to the top-level EGO shared directory (for example, this may be `/share/ego` in Linux/UNIX or `\\HostF\EGOshare` in Windows). If you did not configure management hosts, the top-level EGO directory refers to the EGO installation directory (by default, this is `/opt/ego` in Linux/UNIX, or `C:\EGO` in Windows).

This document uses *`EGO_TOP`* to describe the top-level EGO directory.

Log files

Log files for the reporting services and data loaders are available in the `logs` subdirectory in the PERF directory (*`EGO_TOP`*`/perf/logs`).

There are seven logging levels that determine the detail of messages recorded in the log files. In decreasing level of detail, these are `ALL` (all messages), `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF` (no messages).

By default, all service log files log messages of `INFO` level or higher (that is, all `INFO`, `WARN`, `ERROR`, and `FATAL` messages).

You can change the logging level of the reporting services or the data loaders by editing the `log4j.properties` file.

You can dynamically change the logging level of the `plc` service or the data loaders using the loader controller client tool, but these changes will be lost if you restart the `plc` service.

Event data files

The events logger stores event data in event data files.

The EGO allocation event data file is named `ego.stream` by default and has a default maximum size of 10MB.

When a data file exceeds this size, the events logger archives the file and creates a new data file. The events logger maintains one archive file and overwrites the old archive with the new archive. The event data loaders read both the data files and the archive files.

The default archive file name is `ego.stream.0` for EGO, and the data and archive files are both located in `EGO_TOP/kernel/work/data` by default.

If your system logs a large number of events, you should increase the maximum file size to see more archived event data. If your disk space is insufficient for storing these files, you should decrease the maximum file size, or change the file path to a location with sufficient storage space.

You can manage your event data files by editing the relevant system configuration files. Edit `ego.conf` for the EGO allocation event data file configuration.

Best practices for event data loader tuning

Tuning may help to prevent event data files from switching too quickly to be read by the data loader, resulting in lost records in the data being loaded.

The following factors should be considered when tuning the event data loader:

- stream file size
- loader interval
- performance of the target database
- batch size of the loader
- performance of the host running the loader

1. Stream file size

   This is the most critical factor. For a production cluster, check the stream file switch time. The average switch time should be greater than 3 minutes. If the switch time is too frequent, increase the file size. Note that the file size also impacts the stream file writing speed so if the file is too large, writing speed will slow down.

   The stream file size, specified in Mbytes, is controlled by the parameter EGO_DATA_MAXSIZE in `ego.conf`.
2. Loader interval

   This value, specified in seconds, must be less than the stream file switch time but greater than the database loading speed.

To set the loader interval, open the `plc_ego.xml` file and change the Interval attribute for egoeventsloader. For example:

```
<DataLoader Name="egoeventsloader" Interval="100" Enable="true"
 LoadXML="dataloader/egoevents.xml" />
```

The remaining factors are related to data loading performance and should all be optimized for speed.

If tuning cannot provide satisfactory loader performance, it is possible to create multiple plc service instances for balancing the workload. Contact Platform Professional Services for more information.

Data sources

Data sources

Data sources provide the cluster operation data to the data loaders. Data sources include daemon status files, log files, and event files containing cluster operation data.

Configuration of the data sources

After editing the `plc_service.xml` or `ego.conf` configuration files, restart EGO on the master host (`egosh ego restart` *master_host_name*) for your changes to take effect.

Action	Configuration files	Parameter and syntax
Change the size of the EGO allocation event data files.	`ego.conf`	`EGO_DATA_MAXSIZE =` *max_alloc_file_size* where • *max_file_size* is the maximum size of the allocation event data file before the events logger creates an archive.
Change the location of the EGO allocation event data files.	`ego.conf`	`EGO_DATA_FILE =` *alloc_file_path* where • *alloc_file_path* is the path to the allocation event data file. This includes the name of the allocation event file.

Default behavior of data sources

The EGO allocation event data file is named `ego.stream` and has a maximum size of 10 MB. When a data file exceeds this size, the events logger archives the file and creates a new data file. The events logger only maintains one archive file and overwrites the old archive with the new archive. The archive file is named `ego.stream.0`. The two EGO files are located in `EGO_TOP/kernel/work/data`.

Data source interactions

Sampling data loaders request cluster operation data from the data sources while other data loaders obtain it directly. The data loaders store this data into tables within the relational database containing raw data. Each data loader contains data that is stored in specific tables in the raw database.

Loader controller

Loader controller

The Platform loader controller (`plc`) manages the data loaders by controlling the schedule in which they gather data from the system.

Configuration of the loader controller

After editing the loader controller service configuration file (`plc_service.xml`), restart the `plc` service and EGO on the master host for your changes to take effect.

Action	**Configuration files**	**Parameter and syntax**
Enable automatic startup of the `plc` service. This is the default behavior.	`plc_service.xml` File location: *`EGO_TOP/`*`eservice/esc/conf/services`	`<sc:StartType>AUTOMATIC</sc:StartType>`
Disable automatic startup of the `plc` service.		`<sc:StartType>MANUAL</sc:StartType>`
Specify the default log level of your `plc` log file.	`log4j.properties` File location: *`EGO_TOP/`*`perf/conf`	`log4j.logger.`*domain_name*`.perf.dataloader=`*log_level*`,` *domain_name*`.perf.dataloader` where • *log_level* is the default log level of your loader controller log files. The loader controller only logs messages of the same or lower level of detail as *log_level*. Therefore, if you change the log level to `ERROR`, the loader controller will only log `ERROR` and `FATAL` messages.

Default behavior of the loader controller

The loader controller service starts automatically when the master host starts up.

Loader controller interactions

The loader controller service controls the scheduling of the data loaders. Sampling data loaders request cluster operation data from the data sources while other data loaders obtain it directly. The data loaders store this data into tables within the relational database containing raw data. Each data loader contains data that is stored in specific tables in the raw database.

Data loaders

Data loaders

Data loaders are polling loaders or history data loaders. The data loaders gather data and load this data into specific tables in the relational database containing raw data. Data loaders handle daylight savings automatically by using GMT time when gathering data. Data loaders are controlled by the Platform loader controller (`plc`) service.

Configuration of the data loaders

After editing the loader controller configuration files, restart the `plc` service for your changes to take effect.

Action	**Configuration files**	**Parameter and syntax**
Specify the frequency of data gathering for the specified data loader.	Loader controller configuration files for your data loaders: - `plc_ego.xml` (EGO) - `plc_soam.xml` (Symphony) File location: `*EGO_TOP*/perf/conf/plc`	`<DataLoader Name="*loader_name*" Interval="*gather_interval*" ... />` where - *loader_name* is the name of your data loader - *gather_interval* is the time interval between data gathering, in seconds
Enable data gathering for the specified data loader. This is the default behavior.		`<DataLoader Name="*loader_name*" ... Enable="true" ... />` where - *loader_name* is the name of your data loader
Disable data gathering for the specified data loader.		`<DataLoader Name="*loader_name*" ... Enable="false" ... />` where - *loader_name* is the name of your data loader
Enable data loss protection for the specified data loader. This is the default behavior.	Specific data loader configuration file: *dataloader_name*`.xml` File location: `*EGO_TOP*/perf/conf/dataloader`	`<Writer ... EnableRecover="Y">`
Disable data loss protection for the specified data loader.		`<Writer ... EnableRecover="N">`

Action	Configuration files	Parameter and syntax
Specify the default log level of your data loader log files.	`log4j.properties` File location: *`EGO_TOP`*`/perf/conf`	`log4j.logger.`*`domain_name`*`.perf.dataloader=`*`log_level`*`,`*`domain_name`*`.perf.dataloader` where - *log_level* is the default log level of your data loader log files. The data loaders only log messages of the same or lower level of detail as *log_level*. Therefore, if you change the log level to `ERROR`, the data loaders will only log `ERROR` and `FATAL` messages.

Data gathering methods

The data loaders use different methods of gathering data, depending on the types of data sources from which the data loaders gather data.

Collect	A data collecting loader has full control over what data is gathered from the data sources.
Retrieve	A data retrieving loader does not have full control over what data is gathered and needs to send a request to the data sources. The data sources send the requested set of logs or events back to the data loader.
Sample	A data sampling loader does not have full control over what data is gathered and needs to send a request to the data sources. The data sources send the requested system status information back to the data loader.

Default behavior of data loaders

Data loaders gather data from data sources at regular intervals. The following are lists of the data loaders and default behavior:

Table 1: EGO data loaders

Data loader name	Data type	Data gathering interval	Data loads to	Loader type	Data gathering method

Consumer resource (`egoconsumerresloader`)	resource allocation	5 minutes	`CONSUMER_DEMAND`
			`CONSUMER_RESOURCE_ALLOCATION`
			`CONSUMER_RESOURCELIST`
				polling	sample

Dynamic metric (`egodynamicresloader`)	host-related dynamic metric	5 minutes	`RESOURCE_METRICS` `RESOURCES_RESOURCE_METRICS`	polling	sample

Data loader name	Data type	Data gathering interval	Data loads to	Loader type	Data gathering method
EGO allocation events (`egoeventsloader`)	resource allocation	5 minutes	`ALLOCATION_EVENT`	polling	collect
Static attribute (`egostaticresloader`)	host-related static attribute	1 hour	`ATTRIBUTES_RESOURCE_METRICS` `RESOURCE_ATTRIBUTES`	polling	sample

Table 2: Symphony data loaders

Data loader name	Data type	Data gathering interval	Data loads to	Loader type	Data gathering method
Session (`symsessionloader`)	open and suspended sessions	5 minutes	`SESSION_ATTRIBUTES`	polling	sample
Session history (`symsessionhistloader`)	switched session history	5 minutes	`SESSION_HISTORY`	history	collect
Task history (`symtaskhistloader`)	switched task history	5 minutes	`TASK_ATTRIBUTES`	history	collect

- The session data loader will miss the session data if the session started and finished within the sampling interval.
- The task history data loader will start to collect data within three days before the first start time of the loader controller. The data loader will continue to collect data as long as the loader controller is running.

Data loader interactions

The loader controller service controls the scheduling of the data loaders. Sampling data loaders request cluster operation data from the data sources while other data loaders obtain it directly. The data loaders store this data into tables within the relational database containing raw data. Each data loader contains data that is stored in specific tables in the raw database.

Data purger

Data purger

The data purger (`purger`) service maintains the size of the database by archiving old records and purging them from the database.

Configuration of the data purger

After editing the data purger configuration files, restart the `purger` service for your changes to take effect.

After editing the data purger service configuration file (`purger_service.xml`), restart the `purger` service and EGO on the master host for your changes to take effect.

Action	Configuration files	Parameter and syntax
Specify the default duration of time that the records are stored in the database before being purged.	`purger` configuration files: • `purger_ego.xml` (EGO) • `purger_soam.xml` (Symphony) File location: *`EGO_TOP`*`/perf/conf/purger`	`<TableList Duration="`*expiry_time*`">` where • *expiry_time* is the expiry time, in days, up to a maximum of 31 days This default expiry time applies to all tables, but may be overridden for individual tables by specifying a record expiry time for that table.
Specify the duration of time that the records in a specific table are stored in the database before being purged.		`<Table TableName="`*table_name*`" ... Duration="`*expiry_time*`">` where • *table_name* is the name of the individual table • *expiry_time* is the expiry time, in days, up to a maximum of 31 days This expiry time overrides the default expiry time for all records as specified in the `<TableList>` element.
Disable archiving of old data from a specific table before being purged. This is the default behavior.		`<Table TableName="`*table_name*`" ... Archive="false">` where • *table_name* is the name of the individual table
Enable archiving of old data from a specific table before being purged.		`<Table TableName="`*table_name*`" ... Archive="true">` where • *table_name* is the name of the individual table

Action	Configuration files	Parameter and syntax
Specify the daily time in which the data purger purges old data.	`purger_service.xml` File location: *EGO_TOP*/`eservice/esc/conf/services`	`<ego:Command>...` *file_path*`/purger... -t` *purge_time* where • *purge_time* is the 24-hour daily time, in `hh:mm` format.
Specify the time interval in which the data purger purges old data, starting from when the `purger` service first starts up.		`<ego:Command>...` *file_path*`/purger... -t * [`*purge_time_interval*`]` where • *purge_time_interval* is the time interval, in hours.
Enable automatic startup of the `purger` service. This is the default behavior.		`<sc:StartType>AUTOMATIC</sc:StartType>`
Disable automatic startup of the `purger` service.		`<sc:StartType>MANUAL</sc:StartType>`
Specify the default log level of your `purger` log file.	`log4j.properties` File location: *EGO_TOP*/`perf/conf`	`log4j.logger.`*domain_name*`.perf.purger=`*log_level*`,` *domain_name*`.perf.purger` where • *log_level* is the default log level of your data purger log files. The data purger only logs messages of the same or lower level of detail as *log_level*. Therefore, if you change the log level to `ERROR`, the data purger will only log `ERROR` and `FATAL` messages.

Record archives

By default, the data purger deletes old data. If you enabled data archiving of any table, the data purger archives old data to a file in the *EGO_TOP*`/perf/work/archive` directory. This file is a ZIP file containing a CSV file. The archive file names use the following format:

tablename_date timestamp`.zip`

where

- *tablename* is the name of the table from where the records are archived.
- *date* is the date of the record in `YYYY-MM-DD` format.
- *timestamp* is the 24-hour time stamp of the record in `HH-MM-SS` format.

Note:

There is a space between the date and the time stamp.

For example, `CONSUMER_DEMAND_2007-01-15 22-15-00.zip` is the name of an archive of the records from the `CONSUMER_DEMAND` table at 10:15:00 p.m. on 15 January, 2007.

Default behavior of the data purger

The data purger service starts automatically when the master host starts up. The data purger maintains records for 14 days before purging them at 12:30 a.m. every day. The data purger is a service that starts up automatically whenever EGO starts up.

Data purger interactions

The data purger moves old records from tables in the relational database and archives them.

Managing reports

Disable automatic startup of the reporting services

When disabling the reporting feature, disable automatic startup of the `plc` and `purger` services.

Disable automatic startup of these services by editing their service configuration files (`plc_service.xml` and `purger_service.xml` for the `plc` and `purger` services, respectively).

1. Navigate to the EGO service directory.

   For example,

   Linux/UNIX: **cd *EGO_TOP*/eservice/esc/conf/services**

   Windows: **cd *EGO_TOP*\eservice\esc\conf\services**
2. Edit the service configuration file and change the service type from automatic to manual.

   In the `<sc:StartType>` tag, change the text from `AUTOMATIC` to `MANUAL`.
3. Stop the service that you changed.
4. In the command console, restart EGO on the master host to activate these changes.

   egosh ego restart *master_host_name*

View the status of the loader controller

Use the loader controller client tool to view the status of the loader controller.

1. Launch the loader controller client tool with the `-s` option. The loader controller is located in `EGO_TOP/perf/version_number/bin`.

   - In UNIX, run **plcclient.sh -s**.
   - In Windows, run **plcclient.bat -s**.

Dynamically change the log level of your loader controller log file

Use the loader controller client tool to dynamically change the log level of your `plc` log file if it does not cover enough detail, or covers too much, to suit your needs.

If you restart the `plc` service, the log level of your `plc` log file will be set back to the default level. To retain your new log level, change the default level of your plc log file.

1. Launch the loader controller client tool with the `-l` option. The loader controller is located in `EGO_TOP/perf/version_number/bin`.

   - In UNIX, run **plcclient.sh -l** *log_level*.
   - In Windows, run **plcclient.bat -l** *log_level*.

   In decreasing level of detail, the log levels are `ALL` (for all messages), `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF` (for no messages).

Dynamically change the log level of your data loader log files

Use the loader controller client tool to dynamically change the log level of your individual data loader log files if they do not cover enough detail, or cover too much, to suit your needs.

If you restart the `plc` service, the log level of your data loader log files will be set back to the default level. To retain your new log level, change the default level of your data loader log files.

1. If you are using the default configuration file, launch the loader controller client tool with the `-n` and `-l` options. The loader controller is located in *`EGO_TOP`*`/perf/`*`version_number`*`/bin`.
   - In UNIX, run **plcclient.sh -n** *data_loader_name* **-l** *log_level*.
   - In Windows, run **plcclient.bat -n** *data_loader_name* **-l** *log_level*.

   In decreasing level of detail, the log levels are `ALL` (for all messages), `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF` (for no messages).

   For example, to change the consumer resource data loader log files to the `ERROR` log level:
   - In UNIX,

     ```
     plcclient.sh -n egoconsumerresloader -l ERROR
     ```
   - In Windows,

     ```
     plcclient.bat -n egoconsumerresloader -l ERROR
     ```

Change the default log level of your reporting log files

Change the default log level of your log files if they do not cover enough detail, or cover too much, to suit your needs.

1. Edit *`EGO_TOP`*`/perf/conf/log4j.properties`.
2. Navigate to the section representing the service you want to change, or to the default loader configuration if you want to change the log level of the data loaders, and look for the *log4j.logger.com.platform.perf* variable.

   For example, to change the log level of the data purger log files, navigate to the following section, which is set to the default `INFO` level:

   ```
   # Data purger ("purger") configuration log4j.logger.com.platform.perf.purger=INFO,
   com.platform.perf.purger
   ```
3. Change the *log4j.logger.com.platform.perf* variable to the new logging level.

   In decreasing level of detail, the valid values are `ALL` (for all messages), `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF` (for no messages). The services or data loaders only log messages of the same or lower level of detail as specified by the *log4j.logger.com.platform.perf* variable. Therefore, if you change the log level to `ERROR`, the service or data loaders will only log `ERROR` and `FATAL` messages.

   For example, to change the data purger log files to the `ERROR` log level:

   ```
   # Data purger ("purger") configuration log4j.logger.com.platform.perf.purger=ERROR,
   com.platform.perf.purger
   ```
4. Restart the service that you changed (or the `plc` service if you changed the data loader log level).

Change the disk usage of EGO allocation event data files

If your system logs a large number of events, increase the disk space allocated to the EGO allocation event data files. If your disk space is insufficient, decrease the space allocated to the EGO allocation event data files or move these files to another location.

1. Edit `ego.conf`.

   Windows: `%EGO_CONFDIR%`

   Linux/UNIX: `$EGO_CONFDIR`

   a) To change the size of each EGO allocation event data file, specify or change the EGO_DATA_MAXSIZE parameter.

      `EGO_DATA_MAXSIZE =` *integer*

If unspecified, this is `10` by default. Change this to the new desired file size in MB.

b) To move the files to another location, specify or change the EGO_DATA_FILE parameter.

`EGO_DATA_FILE =` *file_path*

If unspecified, this is *`EGO_TOP`*`/kernel/work/data/ego.stream` by default.

2. In the command console, restart EGO on the master host to activate this change.

   egosh ego restart *master_host_name*

Change the data purger schedule

To reschedule the deletion of old data, change the time in which the data purger deletes the old data.

1. Edit *`EGO_TOP`*`/eservice/esc/conf/services/purger_service.xml`.
2. Navigate to `<ego:Command>` with the -t parameter in the purger script.
   - In UNIX, this is
     ```
     <ego:Command> ... .../purger.sh -t ...
     ```
   - In Windows, this is
     ```
     <ego:Command> ... ...\purger.bat -t ...
     ```

   By default, the data purger is scheduled to delete old data at 12:30am every day.
3. Change the -t parameter in the data purger script to the new time (-t *new_time*).

   You can change the data purger schedule to a specific daily time, or at regular time intervals, in hours, from when the `purger` service first starts up.

   For example, to change the schedule of the data purger:

   - To delete old data at 11:15pm every day:
     ```
     <ego:Command> ... .../purger... -t 23:15
     ```
   - To delete old data every 12 hours from 00:00 of every day:
     ```
     <ego:Command> ... .../purger... -t *[12]
     ```
4. Stop the `purger` service.
5. In the command console, restart EGO on the master host to activate these changes.

   egosh ego restart *master_host_name*

Change the default record expiry time

To reduce or increase the number of records stored in the database, change the duration of time that a record is stored in the database before it is purged. This applies to all tables in the database unless you also specify the record expiry time in a particular table.

1. Edit the `purger` configuration files for your data loaders.
   - For EGO data loaders, edit *`EGO_TOP`*`/perf/conf/purger/purger_ego.xml`.
   - For Symphony data loaders, edit *`EGO_TOP`*`/perf/conf/purger/purger_soam.xml`.
   - For LSF data loaders, edit *`$LSF_TOP`*`/perf/conf/purger/purger_lsf.xml`.
2. In the `<TableList>` tag, edit the Duration attribute to your desired time in days, up to a maximum of 31 days.

   For example, to have the records purged after 7 days:
   ```
   <TableList Duration="7">
   ```

By default, the records are purged after 14 days.

3. Restart the `purger` service.

Change the record expiry time per table

To reduce or increase the number of records stored in the database for a particular table, change the duration of time that a record is stored in the database for this table before it is purged. The duration only applies to this particular table.

1. Edit the `purger` configuration files for your data loaders.
    - For EGO data loaders, edit `EGO_TOP/perf/conf/purger/purger_ego.xml`.
    - For Symphony data loaders, edit `EGO_TOP/perf/conf/purger/purger_soam.xml`.
2. Navigate to the specific `<Table>` tag with the TableName attribute matching the table that you want to change.

    For example:

    ```
    <Table TableName="RESOURCE_METRICS" TimestampColumn="TIME_STAMP" ... />
    ```
3. Add or edit the Duration attribute with your desired time in days, up to a maximum of 31 days.

    For example, to have the records in this table purged after 10 days:

    ```
    <Table TableName="RESOURCE_METRICS" TimestampColumn="TIME_STAMP" Duration="10" ... /
    >
    ```
4. Restart the `purger` service.

Change the frequency of data collection

To change how often the data loaders collect data, change the frequency of data collection per loader.

1. Edit the `plc` configuration files for your data loaders.
    - For EGO data loaders, edit `EGO_TOP/perf/conf/plc/plc_ego.xml`.
    - For Symphony data loaders, edit `EGO_TOP/perf/conf/plc/plc_soam.xml`.
2. Navigate to the specific `<DataLoader>` tag with the Name attribute matching the data loader that you want to change.

    For example:

    ```
    <DataLoader Name="egodynamicresloader" Interval="300" ... />
    ```
3. Add or edit the Interval attribute with your desired time in seconds.

    For example, to have this plug-in collect data every 200 seconds:

    ```
    <DataLoader Name="egodynamicresloader" Interval="200" ... />
    ```
4. Restart the `plc` service.

Disable data collection for individual data loaders

To reduce unwanted data from being logged in the database, disable data collection for individual data loaders.

1. Edit the `plc` configuration files for your data loaders.
    - For EGO data loaders, edit `EGO_TOP/perf/conf/plc/plc_ego.xml`.
    - For Symphony data loaders, edit `EGO_TOP/perf/conf/plc/plc_soam.xml`.
2. Navigate to the specific `<DataLoader>` tag with the Name attribute matching the data loader that you want to disable.

For example:

```
<DataLoader Name="egodynamicresloader" ... Enable="true" .../>
```

3. Edit the Enable attribute to `"false"`.

   For example, to disable data collection for this plug-in:

   ```
   <DataLoader Name="egodynamicresloader" ... Enable="false" ... />
   ```

4. Restart the `plc` service.

Test the reporting feature

Verify that components of the reporting feature are functioning properly.

1. Check that the reporting services are running.
   a) In the Console, under the Service View of the Cluster Health Dashboard, check that the `State` of the `plc` and `purger` services are `STARTED`.
   b) If you are running the Derby demo database, check that the `State` of the `derbydb` service is `STARTED`.
2. Check that there are no error messages in the reporting logs.
   a) View the `EGO_TOP/perf/logs/plc.`*host_name*`.log` file.
   b) Verify that there are no ERROR messages and that, in the `DataLoader Statistics` section, there are data loader statistics messages for the data loaders in the last hour.

      You need to find statistics messages for the following data loaders:

      - `egoconsumerresloader`
      - `egodynamicresloader`
      - `egoeventsloader`
      - `egostaticresloader`
      - `symsessionloader`
      - `symsessionhistloader`
      - `symtaskhistloader`

   c) View the data purger and data loader log files and verify that there are no ERROR messages in these files.

      You need to view the following log files:

      - `EGO_TOP/perf/logs/dataloader/egoconsumerresloader.`*host_name*`.log`
      - `EGO_TOP/perf/logs/dataloader/egodynamicresloader.`*host_name*`.log`
      - `EGO_TOP/perf/logs/dataloader/egoeventsloader.`*host_name*`.log`
      - `EGO_TOP/perf/logs/dataloader/egostaticresloader.`*host_name*`.log`
      - `EGO_TOP/perf/logs/purger.`*host_name*`.log`
      - `EGO_TOP/perf/logs/dataloader/symsessionloader.`*host_name*`.log`
      - `EGO_TOP/perf/logs/dataloader/symsessionhistloader.`*host_name*`.log`
      - `EGO_TOP/perf/logs/dataloader/symtaskhistloader.`*host_name*`.log`
3. Check the report output.
   a) Produce a standard report.
   b) Verify that the standard report produces a chart or table with data for your cluster.

If you were not able to verify that these components are functioning properly, identify the cause of these problems and correct them.

Disable the reporting feature

You must have root or cluster administrator access on the master host.

1. Disable the EGO allocation events data logging.
   a) Define or edit the EGO_DATA_ENABLE parameter in the `ego.conf` file to disable data logging.

   ```
   EGO_DATA_ENABLE = N
   ```

   b) In the command console, restart EGO on the master host to activate these changes.

   egosh ego restart *master_host_name*
2. Stop the reporting services.

   Stop the `derbydb` (if you are using the Derby demo database), `plc`, and `purger` services.
3. Disable automatic startup of the reporting services.

   Disable automatic startup of the `derbydb` (if you are using the Derby demo database), `plc`, and `purger` services.

Enable the Derby database

It is not mandatory to configure the Derby database during the installation. If you later find that you need to use the reporting feature and do not have access to a production database, you will need to enable the Derby database.

Note:

The Derby database is not supported for any production clusters.

1. Navigate to the EGO service directory.

   For example,

   Linux/UNIX: **cd *EGO_TOP*/eservice/esc/conf/services**

   Windows: **cd *EGO_TOP*\eservice\esc\conf\services**
2. Edit the `derby_service.xml` service configuration file.
3. In the

   ```
   <ego:ResourceRequirement>
   ```

   tag, replace

   ```
   @EMBEDDED_DB_HOST@
   ```

   with the name of the host in which you intend to run the Derby database.

   For example, if you intend to run the Derby database in `hostM`,

   ```
   <ego:ResourceRequirement>select('hostM')</ego:ResourceRequirement>
   ```
4. Launch the database configuration tool.

   - In UNIX (X-Windows only), run **dbconfig.sh**.
   - In Windows, run **dbconfig**.

   a) In the User ID and Password fields, use

   ```
   app
   ```

   as both the user name and password for the Derby database.

   b) Verify that the correct JDBC driver is selected for the Derby database. It should be org.apache.derby.jdbc.ClientDriver

   c) Verify the database URL. It should resemble the following: jdbc:derby://*hostname*:1527/app:

   d) Click Test to test the database connection.

   e) Click OK to save the configuration and exit the utility.

5. In the command console, stop the Derby service.

   egosh service stop derbydb

   Note:

   This step is necessary because the `derbydb` service may run even without explicitly enabling the Derby database.

6. In the command console, restart EGO on the master host to activate these changes.

   egosh ego restart *master_host_name*

Moving to a production database for Symphony

The Derby demo database is not supported for any production clusters. To produce regular reports for a production cluster, you must use a supported commercial database.

The reporting feature supports Oracle 9i, Oracle 10g, and Microsoft SQL Server databases for production clusters.

Move to a production database

- You have a user name, password, and URL to access the database server.
- Your database server account has access to create triggers, sequences, tables, and stored procedures.
- There is appropriate space in the database allocated for the reporting feature.
- Download the latest JDBC driver for the commercial database.
  - The JDBC driver for an Oracle database (`ojdbc14.jar` or newer) is available from the following URL:

    http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
  - The JDBC driver for SQL Server (`jdbc-sqlserver.jar`) is available from the following URL:

    http://msdn2.microsoft.com/en-us/data/aa937724.aspx

1. To create the SOAM and EGO database schemas, change to the applicable schema directory on the master host and run the script. Refer to the following table for the location of schema directories.

   For example, for Oracle,

Database Schema	O/S	Location
SOAM	Linux/UNIX	*EGO_TOP*/perf/soam/*version_number*/DBschema/Oracle
	Windows	*EGO_TOP*\perf\soam\version_number\DBschema\Oracle
EGO	Linux/UNIX	*EGO_TOP*/perf/ego/*version_number*/DBschema/Oracle
	Windows	*EGO_TOP*\perf\ego\version_number\DBschema\Oracle

- For Oracle databases, use `sqlplus` to run the scripts.

  EGO database schema:

  sqlplus *user_name***/***password*@*connect_string* **@egodata.sql** *data_tablespace index_tablespace*

  SOAM database schema:

  sqlplus *user_name***/***password*@*connect_string* **@soamdata.sql** *data_tablespace index_tablespace*

  where

  - *user_name* is the user name on the database.
  - *password* is the password for this user name on the database.
  - *connect_string* is the named SQLNet connection for this database.
  - *data_tablespace* is the name of the tablespace where you intend to store the table schema.
  - *index_tablespace* is the index tablespace of the reporting database.
- For SQL Server databases, use `osql` to run the scripts.

  osql -U *user_name* **-P** *password* **-d** *db_name* **-i** *@update_script* where

- *user_name* is the user name on the database
- *password* is the password for this user name on the database
- *db_name* is the name of the reporting database
- *update_script* is the name of the script

2. Stop the reporting services.

   Stop the `derbydb` (if you are using the Derby demo database), `plc`, and `purger` services.

   egosh service stop purger plc derbydb
3. If you are using the Derby demo database, disable automatic startup of the `derbydb` service; refer to *Disable automatic startup of the reporting services* on page 84.
4. Copy the JDBC driver (which you downloaded) into the PERF and GUI library directories on all management hosts that are expected to run PERF and PMC services.

   You need to copy the JDBC driver to the following directories:

   - Linux/UNIX:
     - `EGO_TOP/perf/version_number/lib`
     - `EGO_TOP/gui/version_number/tomcat/common/lib`
   - Windows:
     - `EGO_TOP\perf\version_number\lib`
     - `EGO_TOP\gui\version_number\tomcat\common\lib`
5. Configure your database connection.
   a) Launch the database configuration tool.
      - In UNIX (X-Windows only), run ***EGO_TOP*/perf/*version_number*/bin/dbconfig.sh**.
      - In Windows, run ***EGO_TOP*\perf\version_number\bin\dbconfig.bat**.

   b) In the User ID and Password fields, specify the user account name and password with which to connect to the database and to create your database tablespaces.

      Note:

      This user account must have been defined in your database application, and must have read and write access to the database tables.

   c) In the JDBC driver field, select the driver for your commercial database.

   d) In the JDBC URL field, enter the URL for your database.

      This should be similar to the format given in Example URL format.

   e) In the Maximum connections field, specify the maximum allowed number of concurrent connections to the database server.

      This is the maximum number of users who can produce reports at the same time.
6. Restart the reporting services.

   egosh service start plc purger
7. Restart the Platform Management Console.

   Note:

The Platform Management Console will be unavailable during this step.

a) In the command console, restart the `WEBGUI` service.

   egosh service stop WEBGUI

   egosh service start WEBGUI

The report data will now be loaded into the production database and the Console will use the data in this database. Note that some of your custom reports may not be compatible with the production database if you used non-standard SQL code.

PART

II

Resource Sharing and Distribution

CHAPTER

6

Resource Groups

Understanding resource groups

Resource groups overview

Resource groups organize a heterogeneous resource pool.

Resource groups are logical groups of hosts. Resource groups provide a simple way of organizing and grouping resources (hosts) for convenience; instead of creating policies for individual resources, you can create and apply them to an entire group. Groups can be made of resources that satisfy a specific static requirement in terms of OS, memory, swap space, CPU factor, and so on, or that are explicitly listed by name.

The cluster administrator can define multiple resource groups, assign them to consumers, and configure a distinct resource plan for each group. For example:

- Define multiple resource groups: A major benefit in defining resource groups is the flexibility to group your resources based on attributes that you specify. For example, if you run workload units or use applications that need a Linux OS with not less than 1000 MB of maximum memory, then you can create a resource group that only includes resources meeting those requirements.

  Note:

  No hosts should overlap between resource groups. Resource groups are used to plan resource distribution in your resource plan. Having overlaps causes the hosts to be double-counted (or more) in the resource plan, resulting in recurring under-allocation of some consumers.
- Configure a resource plan based on individual resource groups: Tailoring the resource plan for each resource group requires you to complete several steps. These include adding the resource group to each desired top-level consumer (thereby making the resource group available for other sub-consumers within the branch), along with configuring ownership, enabling lending/borrowing, specifying share limits and share ratio, and assigning a consumer rank within the resource plan.

Resource groups are either specified by host name or by resource requirement using the select string.

Tip:

You can use the command `egoconfig addresourceattr` to add a custom tag to any hosts and then specify that tag when creating a resource group. See the reference for more information.

By default, EGO comes configured with three resource groups: InternalResourceGroup, ManagementHosts, and ComputeHosts.

InternalResourceGroup and ManagementHosts should be left untouched, but ComputeHosts can be kept, modified, or deleted as required.

Note:

Requirements for resource-aware allocation policies (where only certain resources that meet specified requirements are allocated to a consumer) can be met by grouping resources with common features and configuring them as special resource groups with their own resource plans.

Notes on setting CPU slots

When you create a resource group in the Platform Management Console, you must decide on how many slots to assign to each host. The assignment of slots to hosts is a critical function that serves to match the host's resources with the expected workload. If a host is too heavily loaded, performance suffers. If it is underutilized, resources are wasted. Generally, as a starting point, one slot per CPU is allocated for each service instance.

Once the number of slots per host has been configured, it is suggested to monitor host loading via the Platform Management Console. If an adjustment is required, reconfigure the number of slots per host. If hosts are overburdened, decrease the number of slots that are assigned to them. Conversely, if hosts are underutilized, add more slots to them.

Note the following:

- There is a 1-to-1 mapping between small workload units (for example, a session, a task, etc.) and slots.
- If there is a differing individual host value of “slots per host,” it overrides the setting of *x* slots per host you set for the resource group. The host level setting overrides the group level configuration.

  For example, if there are 10 hosts in a resource group and you choose in the Console to set up 5 slots per host, you would normally expect to see 50 slots listed within the Member Host Summary section of a resource group's properties page. However, if you see a different number showing in the summary (for example, 45), then an administrator has manually overridden the settings for one or more hosts. This individual value overrides the group setting configured in the Console.

  In some cases, even if an administrator has not manually changed the slots per host" value, you may still see an unexpected number in the Member Hosts Summary section. This may mean that certain hosts within this particular resource group are double-allocated, meaning they are allocated to more than one resource group. In cases of double-allocation, the sum of the allocated slots displays in the Member Hosts Summary section, not the number of slots for this resource group alone. It is advised not to double-allocate slots.
- If you want to change the value of the number of slots per CPU, it must be specified on the workload management side (outside of EGO).
- The value for the number of CPUs per host is automatically detected during installation.
- The number of slots per host can be defined in the Platform Management Console as an expression. Here are the expression guidelines:
  1. All valid resource requirement expressions are supported, for example, (a*b), (a/b), (a+b), where a and b are resource names or integer/decimal values .
  2. The resource can be one of the following types of static resource:
     - host type defined in the HostType section of ego.shared file. The type is evaluated to 1 if it matches the host, 0 otherwise.
     - nprocs
     - ncores
     - maxmem

Lost-and-found resource groups

When host slots are allocated to a client, the vemkd detects the resource group to which the host belongs. But when the vemkd restarts, there is a brief interval (while host information gets updated) where it may not immediately detect the host's resource group. It is during this update interval that Platform EGO creates a temporary resource group called “LOST_AND_FOUND”. The vemkd adds any host with a current allocation to this resource group if it cannot immediately detect an assigned group. Once vemkd completes its update of host information and detects the host's assigned resource group, the host automatically rejoins it.

Note:

This only happens if the host is already allocated and vemkd must trace its resource group. If the host does not currently belong to an allocation, then vemkd does not perform a search for a resource group.

Similarly, if a host with allocated slots is permanently removed from its resource group (thus never rejoining its original resource group when vemkd restarts), the vemkd adds this host to the “LOST_AND_FOUND” group. It will remain in this group until the cluster administrator frees up the allocation on the host.

Static resources

Static resources are built-in resources that represent host information that does not change over time, such as the maximum memory 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.

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

Note:

You can use the command `egoconfig addresourceattr` to add a custom tag to any hosts and then specify that tag when creating a resource group. See the reference for more information.

Index	Measures	Units	Determined by
scvgf	scavenging flag	string	configuration
type	host type	string	configuration
model	host model	string	configuration
hname	host name	string	configuration
cpuf	CPU factor	relative	configuration
ncpus	number of processors	processors	lim
nprocs	number of processors	processors	LIM
ncores	number of cores	cores	LIM
nthreads	number of threads	threads	LIM
ndisks	number of local disks	disks	lim
maxmem	maximum memory	MB	lim
maxswp	maximum swap space	MB	lim
maxtmp	maximum space in /tmp (Linux/UNIX) or OS default temp directory (Windows)	MB	lim

Scavenging flag (scvgf)

Scavenging flag is a configurable external attribute assigned to a host, identifying that it is available for scavenging. Can be turned on or off.

Host type (type)

Host type is a combination of operating system and CPU architecture. All computers that run the same operating system on the same computer architecture are of the same type. You can add custom host types in the HostType section of `ego.shared`. This alphanumeric value can be up to 29 characters long.

An example of host type is LINUX86.

Host model (model)

Host model is the combination of host type and CPU speed (CPU factor) of your machine. All hosts of the same relative type and speed are assigned the same host model. You can add custom host models in the HostModel section of `ego.shared`. This alphanumeric value can be up to 29 characters long.

An example of host model is Intel_IA64.

Host name (hname)

Host name specifies the name with which the host identifies itself.

CPU factor (cpuf)

CPU factor (frequently shortened to cpuf) is a value representing the speed of the host's CPU relative to other hosts in the cluster. For example, if one processor is twice the speed of another, its CPU factor should be twice as large. For multiprocessor hosts, the CPU factor is the speed of a single processor.

The CPU factors are detected automatically or defined by the administrator.

Number of CPUs (ncpus)

By default, the number of CPUs represents the number of cores.

The number of CPUs can be defined by the cluster administrator (either globally or per-host) to consider one of the following:

- processors
- processors and cores
- processors, cores, and threads

Globally, this definition is controlled by the parameter EGO_DEFINE_NCPUS in `ego.conf` (shared directory). The default behavior for ncpus is to consider the number of cores (EGO_DEFINE_NCPUS=**cores**).

Note:

On a machine running AIX, ncpus detection is different. Under AIX, the number of detected physical processors is always 1, whereas the number of detected cores is the number of cores across all physical processors. Thread detection is the same as other operating systems (the number of threads per core).

Number of processors

The number of physical processors (sockets, for example).

Number of cores

The number of cores (per processor) * the number of processors.

Number of threads

The number of threads per CPU core.

Number of disks (ndisks)

The number of disks specifies the number of disks a machine has.

Maximum memory (maxmem)

Maximum memory is the total available memory of a machine, measured in megabytes (MB).

Maximum swap (maxswp)

Maximum swap is the total available swap space a machine has, measured in megabytes (MB).

Maximum temporary space (maxtmp)

Maximum temporary space is the total temporary space a machine has, measured in megabytes (MB).

About the selection string

The selection string is used to specify resource requirements when creating or modifying resource groups. Resource requirements are properties of a host.

The selection string can be augmented using order so you can specify which selection resources are the most important.

Defining the number of slots per host

Symphony offers you the ability to specify a number of slots for each host in a resource group that is tailored to the individual host’s attributes. You can take advantage of this flexibility when you want to configure a different number of slots for each host in a group that has heterogeneous hosts. Heterogeneous hosts are defined as hosts having a non-uniform number of processors, cores, etc within a resource group or cluster. You can define the number of slots based on resource attributes (for example, the number of processors per host or the amount of maximum memory per host), which might differ among hosts. Without this ability, a user must list all of the hosts one by one in the resource group in order to set the number of slots per host in a heterogeneous cluster, which could reduce the cluster’s expandability and increase the burden of managing the cluster.

The number of slots per host can be defined by the cluster administrator to consider one of the following host attributes:

- number of CPUs (By default, the number of CPUs = number of cores (per processor) * the number of processors. However, depending on the configuration of parameter EGO_DEFINE_NCPUS in ego.conf, the number of CPUs can equate to other CPU resources such as the number of processors or threads.)
- number of processors
- number of cores
- maximum memory

Symphony allows you to configure the host attribute as a factor in a simple expression to derive a new value for host slots. For example, the number of host slots is equal to the number of cores multiplied by two.

For enhanced flexibility, Symphony also supports the use of advanced expressions to define the number of slots per host. For example, you can define the number of slots based on the type of host operating system. For a description of expression guidelines, refer to

Defining the number of slots per host can be achieved by configuration via the Platform Management Console.

Create a resource group by host names

You must be logged in as a cluster administrator and you should have already added most of your hosts to the cluster.

Create new resource groups from the Platform Management Console to ensure your consumers have the appropriate group of compute hosts available to them. Resource groups are often the easiest way to create a homogeneous group of hosts for a consumer (for example, all Linux machines). You can create a resource group by resource requirement (dynamic) or by host names (static). This procedure creates a resource group by host names.

Remember:

When you create a resource group by host names, you select specific member hosts. If any new hosts are added to the cluster, they need to be manually added to a resource group.

1. In the Platform Management Console, click Resources > Configure Resource Groups.

   Any existing resource groups are listed.
2. Select Global Actions > Create a Resource Group.
3. Specify a resource group name.

   Resource group names must consist of letters and numbers only (no spaces or special characters) and must be 64 characters or less.
4. (Optional) Include a description of the resource group.

   There is a 200-character maximum.
5. Specify how many slots per host you would like to have the system count.

   Select the following expression: Number of slots per host is equal to Number of CPUs * **1**.

   For expression guidelines, refer to *Notes on setting CPU slots*. The maximum number of slots per host is 9999.
6. For the Resource Selection Method, select Static (List of Names).

   Static resource selection means that you are selecting specific hosts to belong to this resource group.
7. Under Filter display of member hosts > Hosts to Show in List, select how you would like to filter your host list.

   - All hosts gives you a list of all hosts that belong to your cluster. You cannot specify any resource requirements.
   - Hosts filtered by resource requirement lets you filter your hosts and display a list of candidates for your resource group based on a set of resource requirements. For example, you can specify all hosts that are Linux.
8. If you chose to filter hosts by resource requirement, specify the resource requirement you want.

   For example, **select(LINUX86)**.

   Note the following:

   - The entered expression gets evaluated against each host in the cluster. If a host is found to satisfy the stated resource requirement (if it returns true/ non-zero), then the host is added to the host group.
   - Use the syntax from the selection string to specify your resource requirements. You do not need to use XML equivalents in the Platform Management Console.

Resource Requirement	Description	Example
maxmem	The maximum RAM available	select(maxmem>400)
maxswp	The maximum swap space	select(maxswp>600)
maxtmp	The maximum temporary space	select(maxtmp>100)
ncpus	The number of CPUs	select(ncpus==1)
type	The type of host	select(LINUX86)
ndisks	Number of disks	select(ndisks>1)

- Ensure that you enter a resource requirement expression that relates to the host and that can be used during evaluation (for example, memory requirement, swap space, temporary disk space, etc.).
- If you specify a Windows host name, it must be the full name not the short name.

Tip:

You can use the command `egoconfig addresourceattr` to add a custom tag to any hosts and then specify that tag when creating a resource group. See the reference for more information.

9. If you have specified a resource requirement or modified one, click Refresh Host List to get an accurate list of hosts below.
10. Expand the Member hosts section if necessary and review the hosts found.

    If you selected to filter hosts by All hosts, the list of hosts provided is all the hosts in your cluster. If you selected Hosts filtered by resource requirement, a list of hosts that currently fulfill the requirements you specified in the resource requirement string section display.
11. Review your member hosts and select the hosts you want using the checkboxes.

    If your host list is long, it may go on for several pages. You can select hosts and click Create at any time and then add more hosts from other pages. Make sure you save before navigating to another page.

    Once you have selected a member host, you can filter the list again with a different resource requirement. The hosts highlighted and check marked are your member hosts.

    By default, if you select no member hosts, all hosts in your cluster are added to this resource group when you create it. Furthermore, if you do not select a host, the resource group type switches from Static to Dynamic.
12. Click Check for overlaps.

    No hosts should overlap between resource groups. Resource groups are used to plan resource distribution in your resource plan. Having overlaps causes the hosts to be double-counted (or more) in the resource plan, resulting in recurring under-allocation of some consumers.

    If any hosts overlap, remove them from this resource group or remove them from the overlapping resource group. The exception is with hosts listed in InternalResourceGroup—although all hosts in the cluster are listed here they are not “double-counted” in the resource plan.
13. Click Create.

To get back to your list of resource groups, click Resource Groups on the top of your page or Cancel at the bottom. You now need to update your resource plan based on this new resource group. Note that the number of slots available for your new resource group in the resource plan is automatically detected from what you specified in your resource group.

Create a resource group by resource requirement

You must be logged in as a cluster administrator and you should have already added most of your hosts to the cluster.

Create new resource groups from the Platform Management Console to ensure your consumers have the appropriate group of compute hosts available to them. Often resource groups are the easiest way of creating a homogeneous group of hosts for a consumer (for example, all Linux machines). You can create a resource group by resource requirement (dynamic resource group using static requirements) or by host name (static resource group using a list of hosts). This procedure creates a resource group by resource requirement.

Remember:

When you create a resource group based on resource requirement, you do not select specific member hosts. If new hosts that meet those requirements are later added to the cluster, they are automatically added to the resource group.

Note:

Dynamic resource requirements are not supported. A dynamic resource requirement is one that fluctuates depending on the load on the host.

---

1. In the Platform Management Console, click Resources > Configure Resource Groups.

   Any existing resource groups are listed.
2. Select Global Actions > Create a Resource Group.
3. Specify a resource group name.

   Resource group names must consist of letters and numbers only (no spaces or special characters) and must be 64 characters or less.
4. (Optional) Include a description of the resource group.

   There is a 200 character maximum.
5. Specify how many slots per host you would like to have the system count.

   Select the following expression: Number of slots per host is equal to Number of CPUs * **`1`**.

   For expression guidelines, refer to *Notes on setting CPU slots*. The maximum number of slots per host is 9999.
6. For the Resource Selection Method, leave the default value Dynamic (Requirements).

   Dynamic means that the member hosts of this resource group vary according to which hosts meet that requirement. The requirement itself is static. Some commonly used requirements to create a dynamic resource group include operating system, the number of CPUs, and maximum memory.

   When you create a dynamic resource group, the new hosts added to the cluster that meet the static resource requirement you specify for the resource group are automatically added to the resource group.

   For example, you can specify a dynamic resource group with the (static) requirement of `select (maxmem>400)` and any existing hosts in the cluster that meet this requirement when you create the resource group become members and any new hosts added to the cluster afterwards that meet this requirement are members.

Note:

Dynamic resource requirements are not supported. A dynamic resource requirement is one that fluctuates depending on the load on the host. A dynamic resource group is created with static (unchanging) resource requirements.

7. Under Filter display of member hosts > Hosts to Show in List, select Hosts filtered by resource requirement.

   Do not select All hosts. You need to specify a resource requirement string.

   You specify a resource requirement that members of this group must meet. You can then preview those members in the next step.

8. Specify the resource requirement you want.

   For example, **select(LINUX86)**.

   Note the following:

   - The entered expression gets evaluated against each host in the cluster. If a host is found to satisfy the stated resource requirement (if it returns true/ non-zero), then the host is added to the host group.
   - Use the syntax from the select string to specify your resource requirements. You do not need to use XML equivalents in the Platform Management Console.

Resource Requirement	Description	Example
maxmem	The maximum RAM available	select(maxmem>400)
maxswp	The maximum swap space	select(maxswp>600)
maxtmp	The maximum temporary space	select(maxtmp>100)
ncpus	The number of CPUs	select(ncpus==1)
type	The type of host	select(LINUX86)
ndisks	Number of disks	select(ndisks>1)

   - Ensure that you enter a resource requirement expression that relates to the host and that can be used during evaluation (for example, maximum memory requirement or maximum swap space).
   - If you specify a Windows host name, it must be the full name not the short name.

   Tip:

   You can use the command `egoconfig addresourceattr` to add a custom tag to any hosts and then specify that tag when creating a resource group. See the reference for more information.

   By default, if you specify no resource requirement, all hosts in your cluster are added to this resource group when you create it.

9. Click Refresh Host List and expand the Member hosts section if necessary.

   A preview list of hosts that currently fulfill the requirements you specified in the resource requirement string section displays. This member list is dynamic and reflects the hosts that are currently on your cluster that meet this criteria. If hosts are added or deleted that meet the requirement you specified, they are automatically added or removed from this list.

10. Review your member hosts.

If you do not like the list of hosts found, refine your resource requirement selection string and try again.

11. Click Check for overlaps.

    No hosts should overlap between resource groups. Resource groups are used to plan resource distribution in your resource plan. Having overlaps causes the hosts to be double-counted (or more) in the resource plan, resulting in recurring under-allocation of some consumers.

    If any hosts overlap, remove them from this resource group or remove them from the overlapping resource group. The exception is with hosts listed in InternalResourceGroup—although all hosts in the cluster are listed here they are not “double-counted” in the resource plan.

12. Click Create.

To get back to your list of resource groups, click Resource Groups on the top of your page or Cancel at the bottom. You now need to update your resource plan based on these new resource groups. Note that the number of slots available for your new resource group in the resource plan is automatically detected from what you specified in your resource group.

Modify resource groups

You must be logged in as a cluster administrator.

Modify resource groups so that you can refine your resource allocations used in your resource plan. You may also want to add or remove hosts from resource groups if you have recently added or removed hosts in your cluster.

1. In the Platform Management Console, click Resources > Configure Resource Groups.

   Any existing resource groups are listed.
2. Click the name of the resource group you want to modify.

   The resource group settings display.
3. Change any settings you want.

   If you select Static (List of Names), follow the guidelines in the topic Create a resource group by host names.

   You cannot modify "number of slots per host" for a group with static allocation while any of the group's resources are allocated to consumers.

   If you select Dynamic (Requirements), follow the guidelines in the topic Create a resource group by resource requirement.

   You cannot modify a group with dynamic membership while any of the group's resources are allocated to consumers.
4. When you are satisfied with your settings, click Apply.

Using the Platform Management Console, check that your resource plan is correct considering any changes to the resource groups you just made.

Tutorial: Creating initial resource groups

Goal

Setting up useful resource groups is essential to making full and efficient use of cluster capabilities. Following the steps below, you create resource groups that set aside specific hosts for management duties and divvy up the remainder of your hosts based on maximum memory.

Description

You have created your consumer tree based on your business needs and you have added most of your hosts to your cluster but have not yet set up an extensive resource plan, created new resource groups, or modified default resource groups. You are preparing to customize the plan for your applications and want to divide your hosts by memory as you expect to run varied workload with some requiring not less than 1000 MB of maximum memory and others requiring very little memory at all. You want to ensure the following for your workload:

- They have access to hosts with the necessary amount of maximum memory
- They have no need to wait for appropriate hosts to become available
- The workload that requires very little memory does not get hosts with a large maximum memory

At a glance

1. Plan your groups
2. Check the ManagementHosts resource group
3. Review and modify the master host candidate list
4. Create new dynamic resource groups
5. Create new resource group by host name
6. Modify your resource plan for new resource groups
7. How to grow: Advanced resource groups

Plan your groups

Resource groups overview

Resource groups are logical groups of hosts. Resource groups provide a simple way of organizing and grouping resources (hosts) for convenience; instead of creating policies for individual resources, you can create and apply them to an entire group. Groups can be made of resources that satisfy a specific static requirement in terms of OS, memory, swap space, CPU factor, and so on, or that are explicitly listed by name.

The cluster administrator can define multiple resource groups, assign them to consumers, and configure a distinct resource plan for each group. For example:

- Define multiple resource groups: A major benefit in defining resource groups is the flexibility to group your resources based on attributes that you specify. For example, if you run workload or use applications that need a Linux OS with not less than 1000 MB of maximum memory, then you can create a resource group that only includes resources meeting those requirements.

  Note:

  No hosts should overlap between resource groups. Resource groups are used to plan resource distribution in your resource plan. Having overlaps causes the hosts to be double-counted (or more) in the resource plan, resulting in recurring under-allocation of some consumers.

- Configure a resource plan based on individual resource groups: Tailoring the resource plan for each resource group requires you to complete several steps. These include adding the resource group to each desired top-level consumer (thereby making the resource group available for other sub-consumers within the branch), along with configuring ownership, enabling lending/borrowing, specifying share limits and share ratio, and assigning a consumer rank within the resource plan.

Resource groups generally fall into one of three categories:

- Resource groups that include compute hosts with certain identifiable attributes a consumer may require in a requested resource (for example, resources with large amounts of memory; considered “dynamic”—new hosts added to the cluster that meet the requirements are automatically added to the resource group)
- Resource groups that only include certain compute hosts (for example, so that specified resources are accessed by approved consumers; considered “static”—any new hosts added to the cluster have to be manually added to the resource group)
- Resource groups that encompass management hosts only (reserved for running services, not a distributed workload; for example, the out-of-the-box “ManagementHosts” group)

Resource groups are either specified by host name or by resource requirement using the select string.

By default, EGO comes configured with three resource groups: InternalResourceGroup, ManagementHosts, and ComputeHosts. InternalResourceGroup and ManagementHosts should be left untouched, but ComputeHosts can be kept, modified, or deleted as required.

Note:

Requirements for resource-aware allocation policies (where only certain resources that meet specified requirements are allocated to a consumer) can be met by grouping resources with common features and configuring them as special resource groups with their own resource plans.

Gather the facts

You need to know which hosts you have reserved as management hosts. You identified these hosts as part of the installation and configuration process. If you want to select different management hosts than the ones you originally chose, you must uninstall and then reinstall EGO on the compute hosts that you now want to designate as management hosts (a master host requires installing the full package), and then run `egoconfig mghost`. The tag `mg` is assigned to the new management host, in order to differentiate it from a compute host. The hosts you identify as management hosts are subsequently added to the `ManagementHosts` resource group.

Management hosts run the essential services that control and maintain your cluster and you therefore need powerful, stable computers that you can dedicate to management duties. Note that management hosts are expected to run only services, not to execute workload.

Ensure that you designate one of your managements host as the master host, and another one or two hosts as failover candidates to the master (the number of failover candidates is up to you, and may depend on the size of your production cluster).

1. Make a list of hosts that have been installed with the full package, and that have the tag `mg` assigned to them (from having run `egoconfig mghost`).

   You should be able to get a list from the person who installed your cluster.

2. Review the list of management hosts.

   Ask yourself if these are your most trusted hosts with the reliability they need to be responsible for the entire cluster.
3. (Optional) Remove any listed management hosts you do not trust.
   a) If you have configured automatic startup during your cluster setup, then run `egoremoverc.sh`.

      Doing this prevents automatic startup when the host reboots, which keeps the host from being re-added dynamically to the cluster.
   b) Run `egoconfig unsetmghost` to remove the host from the management host group.

      Running this command removes the host entry from ego.cluster.*cluster_name*.
   c) If the host is a master candidate, run `egoconfig masterlist` to remove the host from the failover order.
   d) Restart the master host to change the local host from a management host to a compute host, and for the cluster file to get read again.
4. (Optional) Designate different management hosts.
   a) For each Linux/UNIX host you wish to designate as a management host, including master candidates, do the following:
      1. Run the `egoconfig mghost` command:

         egoconfig mghost *EGOshare*

         where *EGOshare* is the shared directory that contains important files such as configuration files to support master host failover (once the `egoconfig mghost` command is run and the files are copied over).

         For example, `egoconfig mghost /share/ego`

         Note that the shared directory is the same for all management hosts.
      2. Set the environment on the local host so that *`EGO_CONFDIR`* gets set properly and the changes take effect.

         Doing this changes *`EGO_CONFDIR`* from a local to shared directory.
      3. Restart the master host so that the cluster file gets read again.
   b) For each Windows host you wish to designate as a management host, including the master candidates, do the following:
      1. Run the `egoconfig mghost` command:

         egoconfig mghost *EGOshare domain_name\user_name password*

         where *EGOshare* is the shared directory that contains important files such as configuration files to support master host failover (once the `egoconfig mghost` command is run and the files are copied over), *user_name* is the egoadmin account, and *password* is the egoadmin password.

         For example, `egoconfig mghost \\Hostx.mycompany.com\EGO\share mycompany.com\egoadmin mypasswd`

         Note:

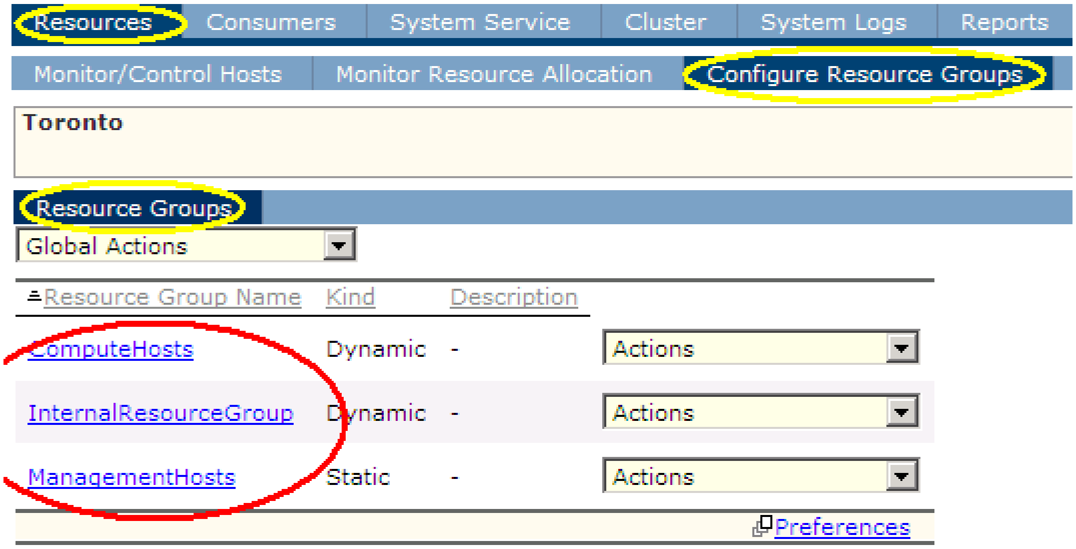
         The shared directory is the same for all management hosts. Also, be sure to use a fully qualified domain name.
      2. Restart the master host so that the cluster file gets read again.

You now have a list of hosts you would like as management hosts. You use this list to check hosts that actually belong to the management hosts resource group.

Recognize the default configurations

To help orient you, here is a list of the default resource groups and resource plan components you see and work with in the Platform Management Console:

- Resource groups:
  - ComputeHosts (executes workload)
  - InternalResourceGroup (runs important EGO components and services)
  - ManagementHosts (runs important EGO components and services)



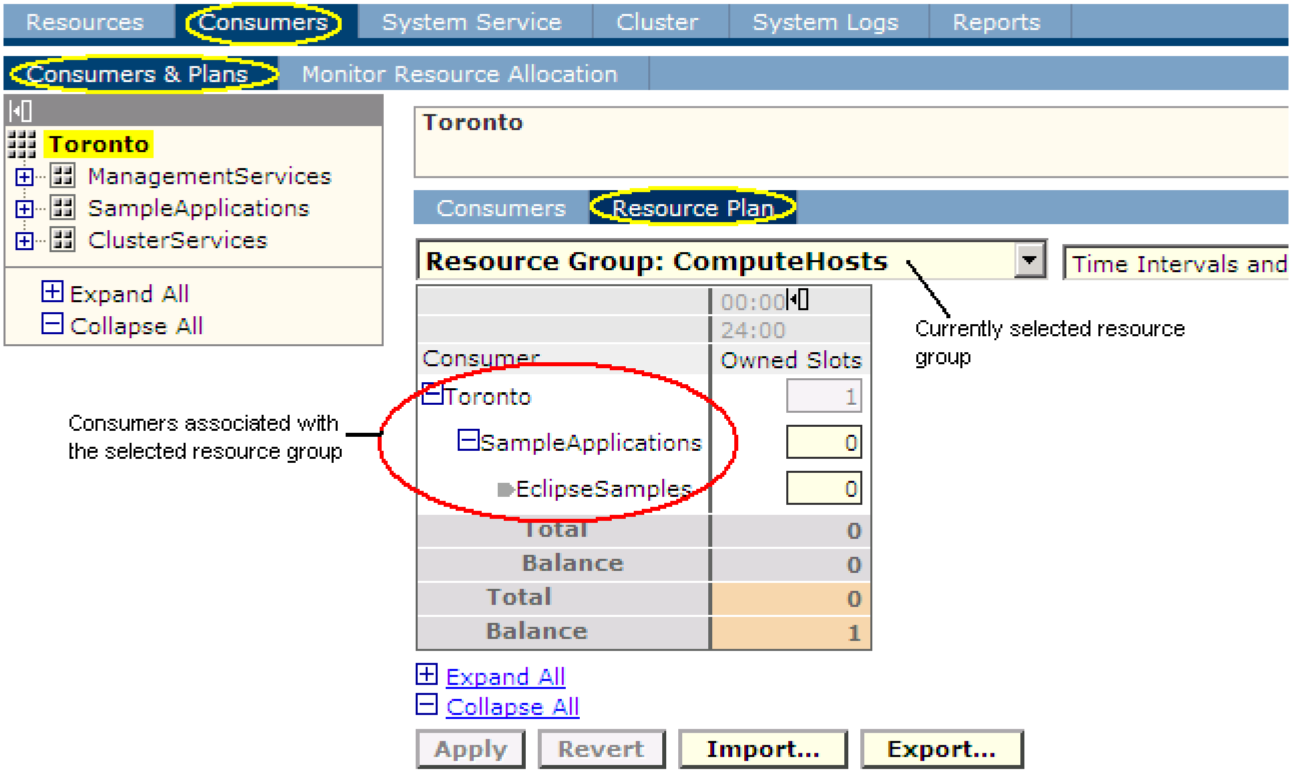
Resource Group Name	Kind	Description
ComputeHosts	Dynamic	-
InternalResourceGroup	Dynamic	-
ManagementHosts	Static	-

Resource Groups

In this tutorial, we work with the ComputeHosts resource group and create new resource groups.

- Resource plan (default resource group upon opening page is ComputeHosts):

  Only consumers registered to a selected resource group show. Select different resource groups to modify corresponding resource plans.



Currently selected resource group

Consumers associated with the selected resource group

In this tutorial, we update the resource plan to include the new resource group you create.

Check the ManagementHosts resource group

You must be logged on to the Platform Management Console as a cluster administrator.

The `ManagementHosts` resource group is created during the installation and configuration process. Each time you install and configure the full package on a host, that host is statically added to the `ManagementHosts` resource group.

You need to ensure that the trusted hosts you identified in the section Gather the facts (above) are the same as the hosts that were configured to be management hosts.

1. From the Platform Management Console, click Resources > Configure Resource Groups > Resource Groups.

   A list of all resource groups displays.

   By default, your resource groups are `ComputeHosts`, `InternalResourceGroup`, and `ManagementHosts`.
2. From the list, click ManagementHosts.

   The properties for `ManagementHosts` display.

   Caution:

   Do not, under any circumstances, modify any of the `ManagementHosts` properties (except for the description). You could seriously damage your cluster.

3. Note and compare the hosts listed in the Member hosts section at the bottom.

   The hosts that are members of the `ManagementHosts` resource group are listed here.

   Do these hosts match the list of hosts you made in the section Gather the facts? If not, contact the person in charge of installation and make sure each management host is configured properly.

   You need the exact host name(s) for the next topic.

You have made sure the hosts you want as management hosts belong to the `ManagementHosts` resource group. The installation and configuration matches your desired cluster setup.

Review and modify the master host candidate list

You must be logged on to the Platform Management Console as a cluster administrator.

Once you have reviewed your `ManagementHosts` resource group, you need to make sure your master host candidate list is correct.

1. Select Cluster > Summary.

   A summary displays.
2. Click Master Candidates.

   The master host is the first host in the list displayed in the right column. Other host names may be listed as candidates or as available hosts (right and left columns, respectively).
3. Review master and candidates.

   The master host is the host listed first in the candidates column. All others under the candidate list should be eligible hosts that are also part of the `ManagementHosts` resource group.
   a) Check the host names against the list you made when you checked the `ManagementHosts` resource group.
   b) Use the controls to move hosts around. Add any hosts that you want as master candidates into the candidates column in the order you want them to failover.

      You cannot remove the master host.

Create new dynamic resource groups

You must be logged on to the Platform Management Console as a cluster administrator. You should not be running any workload while you perform this task because it involves removing an existing resource group.

When you delete a resource group, those hosts are no longer assigned to a consumer. Therefore, you should complete this task before changing your resource plan for the first time. If you have modified the resource plan and want to save those changes, export the resource plan before starting this task.

You can create resource groups that automatically place all your compute hosts in two (or more) different resource groups. You can split your hosts up this way if some of the applications or workload you plan to run on the Symphony cluster have distinct or important memory requirements.

You can logically group hosts into resource groups based on any criteria that you find important to the applications and workload you intend to run. For example, you may wish to distinguish hosts based on OS type or CPU factor.

1. Select Resources > Configure Resource Groups > Resource Groups.

   A list of your existing resource groups displays.

By default, your resource groups are `ComputeHosts`, `InternalResourceGroup`, and `ManagementHosts`.

Caution:

The `InternalResourceGroup`, `ManagementHosts`, and `ComputeHosts` groups should never be deleted. They are special resource groups that contain hosts used for EGO services and out-of-the-box applications.

2. Select Global Actions > Create a Resource Group.

   The resource group properties display.
3. Fill in the resource group properties.
   a) Type a name that describes the hosts that you are going to select for this group. In this example, we use “maxmem_high”.
   b) Do one of the following to define the number of slots per host:

      If th e parameter EGO_ENABLE_SLOTS_EXPR = N in the `ego.conf` file, select 1 slot per CPU; otherwise, define the calculation for the number of slots based on maximum memory in the host:

      1. Choose Number of slots per host is equal to.
      2. Select Maximum Memory from the resource list.
      3. Select / from the list of operators.
      4. Enter **500** in the text box.
   c) Make sure the resource selection method is Dynamic (Requirements).
   d) Under Hosts to Show in List, select Hosts filtered by resource requirement.
   e) In the Resource Requirement String field, type **select(!mg && maxmem > 1000)**.

      The command select ignores any hosts belonging to the `ManagementHosts` resource group (**!mg**) and add any non-management host that has a maximum memory of 1001 MB or more (**maxmem > 1000**).
   f) Click Refresh Host List.

      In the Member hosts section, a list of any hosts (as found in the current cluster) that meet the requirements you specified with the select string is generated.
   g) Review the hosts in the member section and make any modifications you need to the select string until the member list is correct.

      Only hosts that currently match the requirements are displayed here. However, the list is dynamic. As you add hosts to the cluster that meet these requirements, they are automatically added to this resource group.
   h) Click Check for overlaps in the member hosts section to make sure the member hosts do not belong to any other resource groups.

      If you have overlaps, modify your selection string until overlaps no longer exist. Hosts must never overlap between resource groups. Having overlaps causes the hosts to be double-counted (or more) in the resource plan, resulting in recurring under-allocation of some consumers. The exception is with hosts listed in `InternalResourceGroup`—although all hosts in the cluster are listed here, they are not “double-counted” in the resource plan.
   i) Once you have no overlaps, click Create.
4. Click Resource Groups again.

   A list of resource groups displays, including the maxmem_high group you just created.

5. Create a second resource group.

   > **Note:**
   >
   > You can skip this step and go to Create a new resource group by host names instead.

   Follow the same steps above with the following differences.

   a) Name the second resource group “maxmem_low”.
   b) Add the selection string **select(!mg && !(maxmem > 1000))**.

      This resource group is now made up of any compute host not belonging to the `ManagementHosts` resource group and excluding hosts you specified for the maxmem_high resource group.

      We recommend that you specify one resource group that excludes all other resource groups or selection string requirements (specify using "not" (!)). That way, all your hosts fall into one resource group or another.

   You have now deleted the `ComputeHosts` resource group and split all your hosts, except those belonging to the `ManagementHosts` resource group, into two new groups: one made up of hosts with memory over 1000 MB (maxmem_high) and one made up of all other hosts with memory of 1000 MB or less (maxmem_low).

Create a new resource group by host names

If you did not create two resource groups in the following task or did not include all hosts in one of the two resource groups, you can now create a resource group by listing host names.

You must be logged on to the Platform Management Console.

You should have already added most of your hosts to the cluster.

Create a new resource group by host name to include any hosts that may not be already included in a resource group that is dynamic.

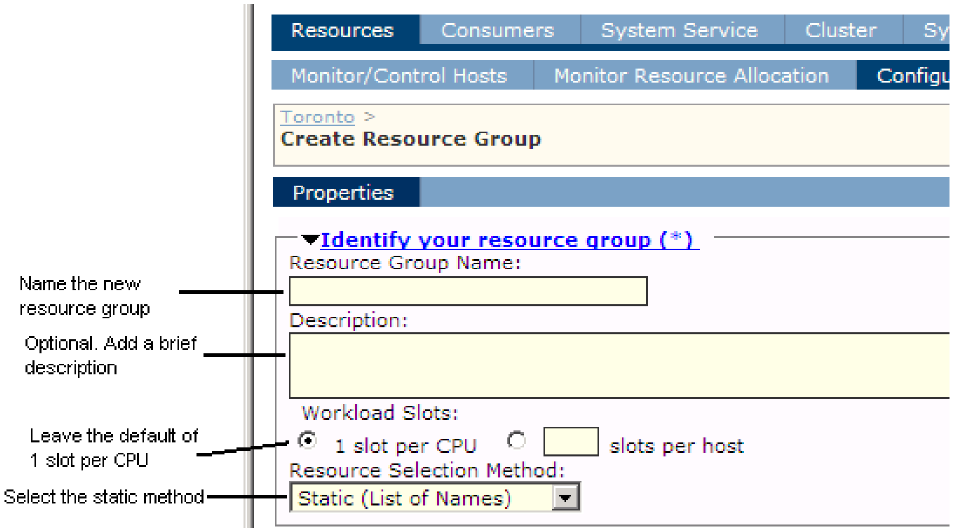
Any new compute hosts that are later added to the cluster, and that you want to add to this resource group, must be manually added.

1. In the Platform Management Console, click Resources > Configure Resource Groups > Resource Groups
2. From the Global Actions drop-down list, select Create a Resource Group.
3. Identify the new resource group in the top section of the Properties page:
   a) Specify a resource group name.

      In this example, we use "my_static".

      Resource group names must consist of letters, numbers, dashes (-), or underscores (_) only (no spaces or special characters) and must be 64 characters or less.
   b) Include a description (max. 200-characters) of the resource group.
   c) Leave the default setting of 1 slot per CPU for Workload Slots (this defines how many slots per host you would like to have the system count; unless you are an advanced user, do not change this setting).
   d) For Resource Selection Method, select Static (List of Names).

      Static resource selection means that you are manually selecting specific hosts to belong to this resource group.



Resource Group Name:

Description:

Workload Slots: ● 1 slot per CPU ○ slots per host

Resource Selection Method: Static (List of Names)

Name the new resource group

Optional. Add a brief description

Leave the default of 1 slot per CPU

Select the static method

4. Under Hosts to Show in List, select All hosts.

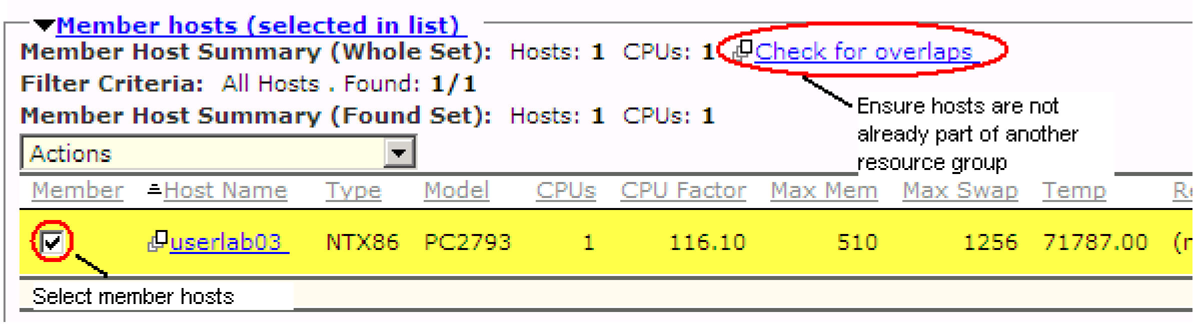
   A list of all hosts that belong to your cluster displays.

5. Review the hosts found in your cluster:
   a) Click Member hosts to expand the section and review the hosts found in your cluster.
   b) Review your member hosts and select the hosts you want using the check boxes.

      If you select no member hosts, all hosts in your cluster are added to this resource group when you create it.
   c) Click Check for overlaps.

      If any hosts overlap, remove them from this resource group or remove them from the overlapping resource group.

      No hosts should overlap between resource groups. Resource groups are used to plan resource distribution in your resource plan. Having overlaps causes the hosts to be double-counted (or more) in the resource plan, resulting in recurring under-allocation of some consumers. The exception is with hosts listed in InternalResourceGroup—although all hosts in the cluster are listed here they are not “double-counted” in the resource plan.



Member Host Summary (Whole Set): Hosts: **1** CPUs: **1**

Filter Criteria: All Hosts . Found: **1/1**

Member Host Summary (Found Set): Hosts: **1** CPUs: **1**

Member	Host Name	Type	Model	CPUs	CPU Factor	Max Mem	Max Swap	Temp	R
☑	userlab03	NTX86	PC2793	1	116.10	510	1256	71787.00	(r

Ensure hosts are not already part of another resource group

Select member hosts

6. Click Create.

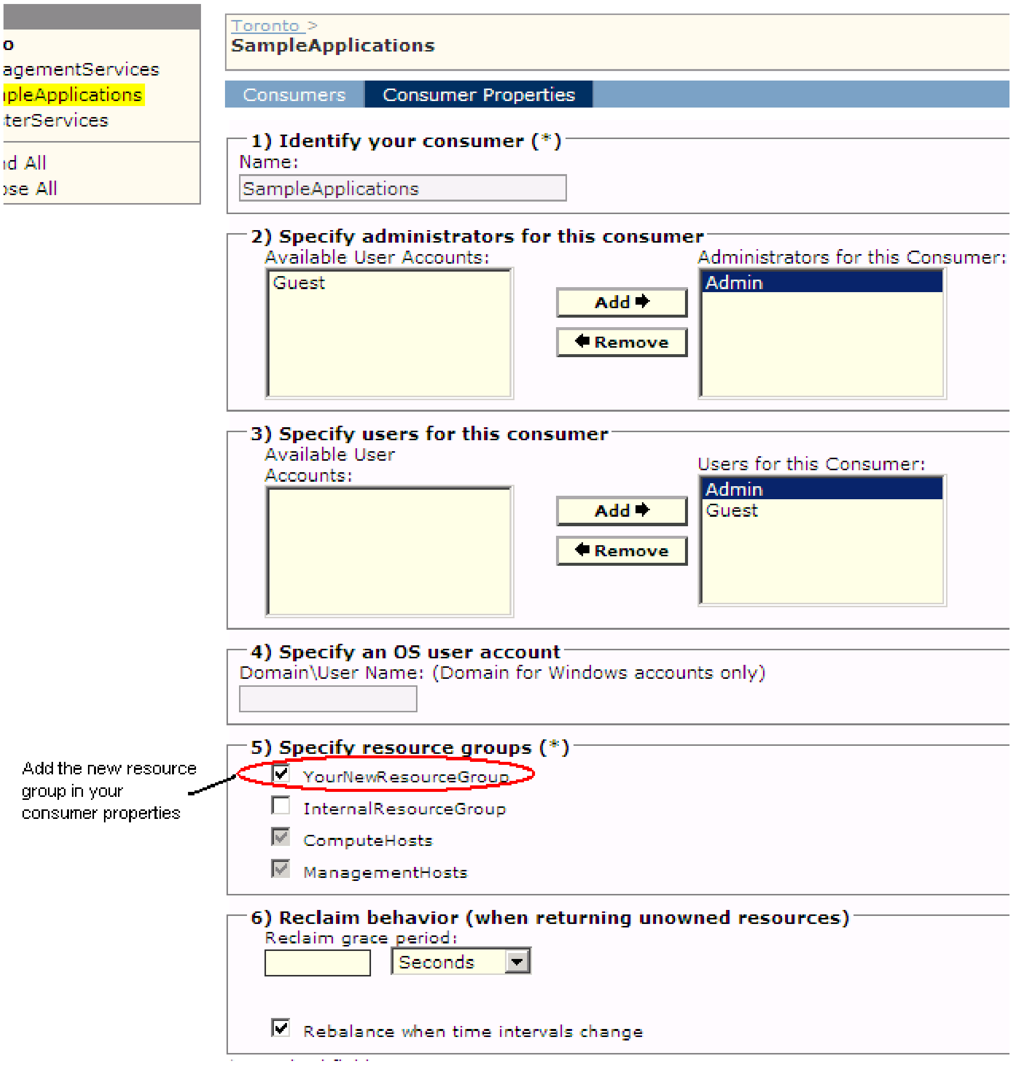
Assign the new resource groups to a consumer

You must have already created the consumers that you want.

You need to assign new resource groups to consumers.

1. Click Consumers > Consumers & Plans > Consumers.
2. Select a consumer to assign the new resource group to.
   - If you have already created your consumers by modifying the out-of-box structure, using the tree, locate and click the consumer to which you want to assign the new resource group.
   - If you have not modified the consumer tree, click SampleApplications from the consumer tree pane on the left to assign the new resource group to this consumer.
3. Click Consumer Properties.
4. Specify one or more resource groups that this consumer should have access to.
5. Click Apply.

   The Consumer Properties page updates and your changes are saved.



SampleApplications

1) Identify your consumer (*)
Name: SampleApplications

2) Specify administrators for this consumer
Available User Accounts: Guest
Administrators for this Consumer: Admin

3) Specify users for this consumer
Available User Accounts:
Users for this Consumer: Admin, Guest

4) Specify an OS user account
Domain\User Name: (Domain for Windows accounts only)

5) Specify resource groups (*)
☑ YourNewResourceGroup
☐ InternalResourceGroup
☑ ComputeHosts
☑ ManagementHosts

6) Reclaim behavior (when returning unowned resources)
Reclaim grace period: Seconds
☑ Rebalance when time intervals change

Add the new resource group in your consumer properties

Modify your resource plan for new resource groups

If you know you intend to create more resource groups, do that first even if you do not know all the details of the resource groups.

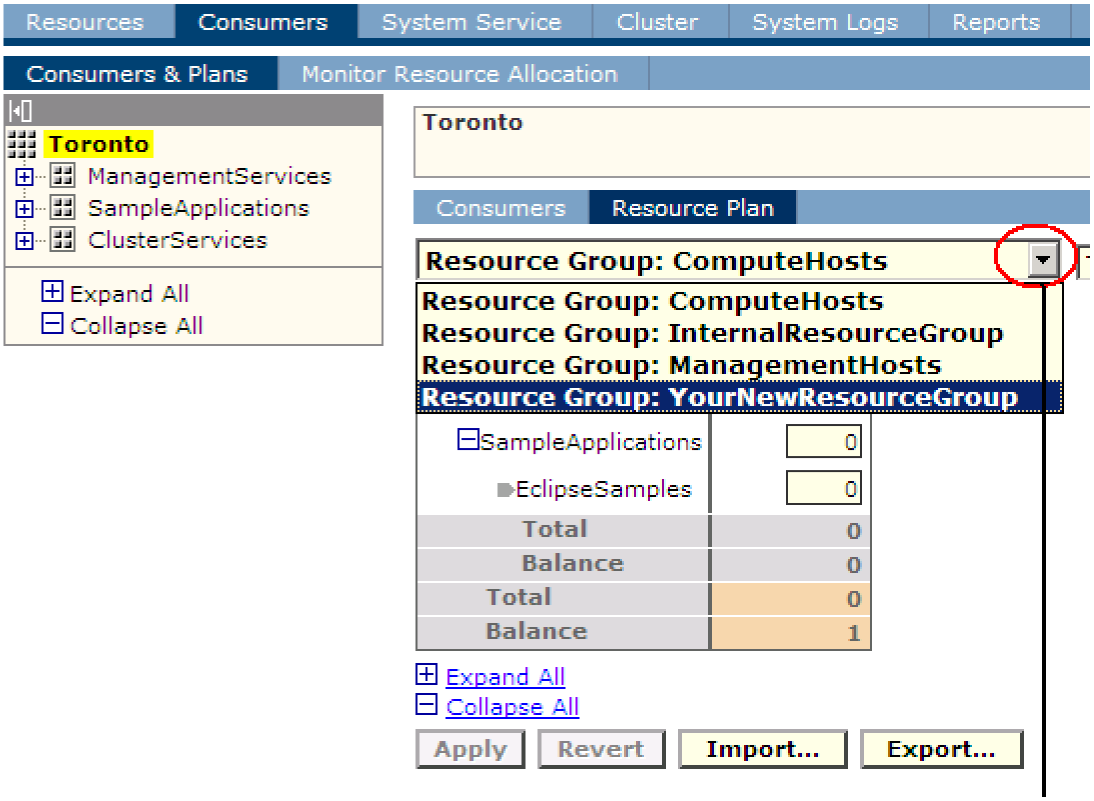
Any time you add, modify, or delete a resource group, you need to manage resource distribution for these resource groups using the resource plan.

1. Click Consumers > Consumers & Plans > Resource Plan.
2. Use the Resource Group drop-down menu to switch between resource groups and modify your resource plan details for each resource group.

Note:

Resources groups that do not yet have consumers assigned to them do not appear in the drop-down menu. Consumers must first be assigned from the Consumers & Plans > Consumers page.

Never make any changes to the `ManagementHosts` resource group in the resource plan.



Resource Group: ComputeHosts

Resource Group: ComputeHosts
Resource Group: InternalResourceGroup
Resource Group: ManagementHosts
Resource Group: YourNewResourceGroup

SampleApplications	0
EclipseSamples	0
Total	0
Balance	0
Total	0
Balance	1

Select your new resource group from the drop-down list

How to grow: Advanced resource groups

Now that you have basic resource groups (one for your management hosts and two or more for your compute hosts) you can begin to specialize and split up one resource group that is based on available memory.

For example, if you know that an application you run requires not only machines with 1001 MB of available memory or more, but also two or more CPUs, you can create a new resource group (and then modify the existing “maxmem_high” resource group) to make these specific resources available to any consumer. The new resource group “maxmemhighmultiCPU” would have the selection string:

select(!mg && maxmem > 1000 && ncpus>=2)

You would then modify the existing resource group “maxmem_high” to read:

select(!mg && !(ncpus>=2) && maxmem > 1000)

As a result, the maxmem_high group uses only single CPU hosts.

CHAPTER

7

About Consumer Trees

About the consumer tree

Overview

The consumer tree organizes consumers into a structure that makes it easy to apply resource plans.

The consumer tree is closely related to the resource plan. The plan cannot be defined without the tree.

The tree only defines organizational relationships among consumers, while the plan defines resource allocation.

The choice of consumers and their hierarchy should reflect long-term business goals because it can be complicated to modify the tree. To make the system adjust to short-term business changes, you can modify the users associated with a consumer, or the resource plans defined in the plan.

Terminology

Component	Description
Tree	The resource distribution tree identifies consumers of the cluster resources, and organizes them into a manageable structure.
Plan (Resource plan)	The resource plan describes the relationship between the consumer tree and resource groups, and defines plans for how cluster resources are to be shared among consumers.
Consumers	A consumer in the tree represents any entity that can demand resources from the cluster. A consumer might be a business service, a business process that is a complex collection of business services, an individual user, or an entire line of business. The consumers ManagementServices, SampleApplications, and ClusterServices, along with their sub-consumers, are installed by default. • ManagementServices has two sub-consumers, EGOManagementServices and SymphonyManagementServices, which run important system services on management hosts in the cluster. Services include derbydb, plc, purger, ServiceDirector, WEBGUI, RS, and WebServiceGateway. ManagementServices is configured to use the ManagementHosts resource group. Do not modify or delete this consumer. • SampleApplications has two sub-consumers, SOASamples and EclipseSamples. The SampleApplications consumer and its sub-consumers can be modified or deleted (although you may want to use the provided samples to begin using Symphony right away). • ClusterServices is configured to use the InternalResourceGroup resource group. It has two sub-consumers, EGOClusterServices and SymphonyClusterServices, which run important system services on every host in the cluster. Do not modify or delete ClusterServices, or use it to run workload units.
Multi-level tree	Company projects are generally structured with multiple layers and components. For example, a project belongs to a department, a department to a business unit and so on. A multi-level consumer tree allows you to configure consumers in a hierarchical fashion to match your business structure. **Note:** As a best practice, restrict the number of tree levels to four.

Component	Description
Tree root	The root of the tree represents the entire cluster and all resources in it. Resources from the root are distributed through the tree to consumers.
Top-level consumers	The consumers attached directly to the root are called top-level consumers. The top-level consumer is the head of a consumer branch.
Leaf consumers	If a consumer has no descendants, it is called a leaf consumer. Applications can only be associated with leaf consumers. Borrow and lend policies are set at this level. **Note:** As a best practice, limit leaf consumers to fewer than 20.
Branches, descendants	If a consumer in the tree has other descendants, thereby creating a branch in a multi-level tree, it is called a branch consumer. Branch consumers exist to redistribute resources down the branch to their descendants. Descendants of a branch consumer may also have descendants, thereby becoming branch consumers themselves. Every branch in the tree ends with a leaf consumer.
Parent	A consumer containing another consumer (a "child"). A parent can contain a child consumer, or be the child of another parent consumer.
Sibling	Two or more consumers sharing the same parent consumer.
Child, sub-consumer	A consumer nested within another consumer (a "parent"). A child (or sub-consumer) of one parent can be the parent to another nested child. A leaf consumer is always a child at the end of the branch.
Consumer administrator	For ease of management, you can create consumer administrators for top-level consumers in a multi-level tree. These users can change the plan for lower-level consumers on their branch (descendants), without requiring cluster administrator permissions. Only a cluster administrator can change the plan for top-level consumers.

Building your consumer tree

Follow these rules when building a consumer tree:

- You must be a cluster administrator.
- You cannot create a consumer beneath a consumer that has anything registered to it. You must unregister before you can create a sub-consumer.
- You can only create five levels of consumers including the cluster (top) level.
- You cannot change your cluster name.
- You must build your tree from the top down.

Defaults

The consumers ManagementServices, SymTesting, SampleApplications, SymExec, and ClusterServices, along with their sub-consumers, are installed by default.

- ManagementServices has two sub-consumers, EGOManagementServices and SymphonyManagementServices, which run important system services on management hosts in the cluster. Services include derbydb, plc, purger, ServiceDirector, WEBGUI, and WebServiceGateway. ManagementServices is configured to use the ManagementHosts resource group. Do not modify or delete this consumer or its sub-consumers.
- SymTesting has one sub-consumer, Symping51, which runs the Symphony configuration testing tool, Symping.
- SampleApplications has two sub-consumers, SOASamples and EclipseSamples.

  The SampleApplications consumer and its sub-consumers can be modified or deleted (although you may want to use the provided samples to begin using EGO right away).
- SymExec has one sub-consumer, SymExec51.

  The SymExec51 consumer lets you submit commands to symexec5.1 using the Run Executable tool.
- ClusterServices is configured to use the InternalResourceGroup resource group.

  It has two sub-consumers, EGOClusterServices and SymphonyClusterServices, which run important system services on every host in the cluster. Do not modify or delete ClusterServices or use it to run workload units.

Next step

Begin to build your tree by adding consumers.

Create a consumer

To create a consumer, you must either be a cluster administrator or a consumer administrator for the branch on which you are creating a consumer.

1. Click Consumers > Consumers & Plans.

   A list of existing top-level consumers in your tree displays.
2. Locate and click on the tree level for which you would like to add a consumer.
3. From Global Actions, select Create a Consumer.

   The Create a Consumer page displays.

   You can create a consumer at any level of your existing tree, except where a consumer already has something registered to it; in this case, no other consumers can be created below it in the tree.
4. Fill in the consumer properties.

   Some of these properties may be already filled out or disabled depending on which tree level you are adding a consumer to.
   a) Specify a name for your new consumer.

      The name can be up to 32 characters long and cannot be the same as any other consumer in your tree.
   b) Choose one or more user accounts to be administrators for this consumer; otherwise, leave this field blank.

      Specified administrators automatically become administrators for any other consumers created on this branch.
   c) Specify zero, one, or more users for this consumer.
   d) Specify the workload execution user account (the OS account under which workload runs).

      This field may be pre-populated, but you can modify it. Windows accounts should include a domain name.

      If you specify a Windows user account that has not already been configured, you have to log on to EGO as the cluster administrator and then run `egosh ego execpasswd` before the execution user can run an activity without exiting.
   e) Specify one or more resource groups this consumer should have access to.

      Only the resource groups specified by this consumer's parent are available for selection. If you have not modified your resource groups, you can keep the default resource group selections.
   f) Specify a reclaim grace period.

      The reclaim grace period is applied when a resource belonging to another consumer is now being reclaimed by its owner consumer. Setting the reclaim grace period higher than the length of time to complete your workload allows workload to finish before the resource is reclaimed. Setting the reclaim grace period to 1 second terminates all workload running and reclaims the resource almost immediately.
   g) (Optional) Check the Rebalance when resource plan changes or time interval changes box.

      If you want EGO to “rebalance” or reset the ownership when a new time interval occurs with a change in ownership of resources, check this box. Similarly, when resources are reclaimed (or passed back to their original owners), you can evoke a rebalancing in accordance with the resource plan.

Before EGO rebalances according to the resource plan, a consumer’s grace period is honored to help ensure workload is completed before being killed.

5. Click Create.

You may need to configure the Windows password of the execution user account.

Configure Windows password

If, when creating or modifying a consumer, you specify a Windows user account that has not already been configured, you have to set the password before you can continue.

Configure the password of every new Windows execution account that you introduce into the system. (The egoadmin account was configured during installation.)

Use the same command to update the system if the execution account password ever changes. You have to register the new password in EGO every time the execution account password changes in Windows.

Restriction:

The password must be 31 characters or less.

1. Log on to a host as egoadmin.
2. Log on to EGO as an EGO authentication user.

   egosh user logon -u *user_name* **-x** *password*

   For example, type

   ```
   egosh user logon -u Admin -x Admin
   ```
3. To configure the Windows password, run

   egosh ego execpasswd -u *user_name* -x *password*

   For example, if the account is mydomain\user2, type

   ```
   egosh ego execpasswd -u mydomain\user2 -x mypasswd
   ```

   Note:

   If you want to set the password for a Windows domain user from a Linux/UNIX host, enclose the Windows domain name in quotation marks:

   ```
   egosh ego execpasswd -u "mydomain\user2" -x mypasswd
   ```

Consumer properties

Property	**Explanation**
Name	A unique name of a department or project in your structure.
Administrators	User accounts or with administrative privileges for this consumer and all consumers beneath it. A consumer administrator can see everything in their branch of the tree, modify any property, assign resources, modify user accounts, and control hosts.
Users	List of user accounts that have access to this consumer without administrative privileges. Users can only view their own properties.
OS user account	The operating system user account under which workload runs. All workload for the consumer runs under the same account, no matter which user submitted workload units. On Linux/UNIX hosts, the `$SOAM_HOME/deploy` directory's group access permissions is set to 7 (rwx). All execution (os) users of consumers must belong to the `deploy` directory owner group. The `deploy` directory is created and owned by the cluster admin os user account (by default, egoadmin). If the consumer runs workload units on a Windows host, include the domain name as shown: *domain_name\user_name* If the consumer runs workload units on a Linux/UNIX host, EGO can automatically strip the domain name from the user name. For example, if you configure the account as mydomain\user2, it is interpreted as mydomain\user2 on Windows but as user2 on Linux/UNIX. Every new Windows execution user account needs to have the password configured with `egosh ego execpasswd`. The egoadmin account is already configured. Any activity started through `egosh` uses the same execution account as configured for the leaf consumer it runs on. The file `ConsumerTrees.xml` contains execution user account information for each registered consumer.
Resource groups	A collection of hosts. You can create resource groups based on similar qualities or by machine names. Only those resource groups assigned to this consumer by its parent consumer are available for assignment.
Reclaim	Reclaim applies to consumers that are borrowing resources. If a client is running workload units on a borrowed resource, you can impose a delay (grace period) so that these units can run uninterrupted before the resource gets returned (reclaimed). Alternately, you can choose to immediately interrupt workload units running on a borrowed resource when it is reclaimed by setting the grace period to 0. When a grace period is left blank or not configured, it defaults to 0 seconds.

Tutorial: Building Your Tree

Goal

Setting up a basic consumer tree that mirrors your business structure vastly increases your ability to manage your resources efficiently. By following the steps below, you gain an understanding of a consumer tree, consumers, and how to plan for the future.

Description

You have recently installed your cluster and have not yet created consumers, modified the resource plan, or modified, removed, or added resource groups. You want to understand what your consumer tree does and how you should build it.

At a glance

1. Gather the facts
2. How your business structure maps to your tree
3. About the consumer tree
4. Recognize the default configurations
5. Create your consumers
6. How you are prepared

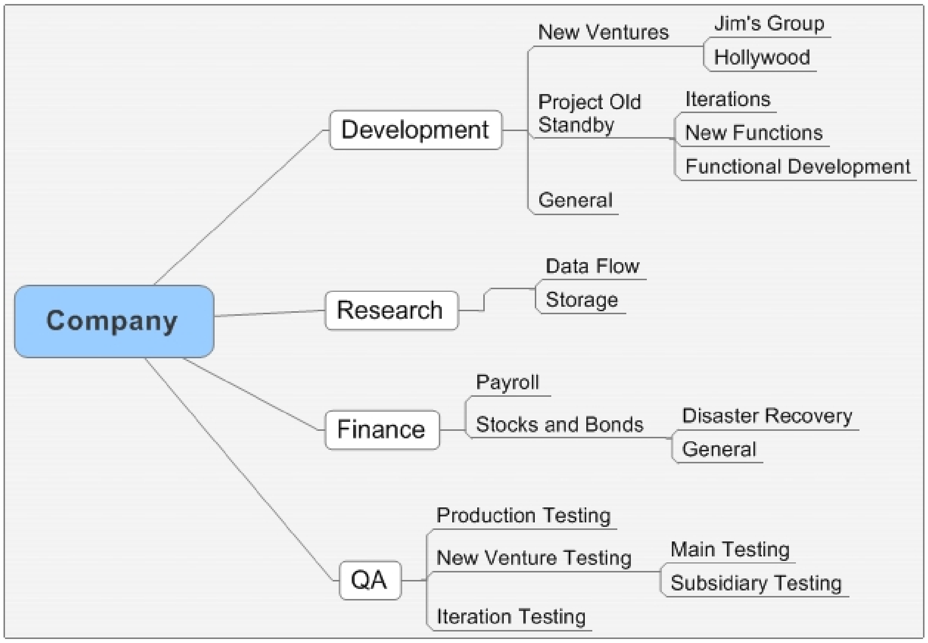
Gather the facts

Before you begin to create your consumers and build your tree, you need to know how you want to control your cluster resources. To begin the process, you need to map out your current business structure.

1. Map out your business structure by hand.

   As a best practice, restrict the number of levels to four.

   For example, your business structure might look like this:



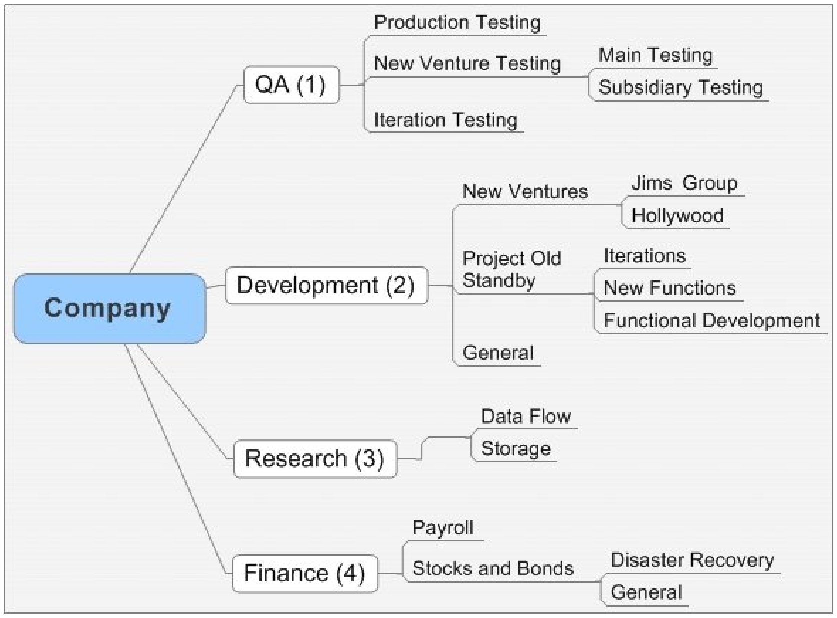
2. Prune your tree.

   Once you have a detailed diagram of your company structure, decide if any branches of your tree do not need to consume cluster resources, and remove them from your diagram.

3. Prioritize your business processes.

   a) For each top-level business process, decide which areas should receive resources first when they need them.

      If I decide that QA should have priority over everything else, and Finance likely needs resources with less urgency, I might set the order like this:

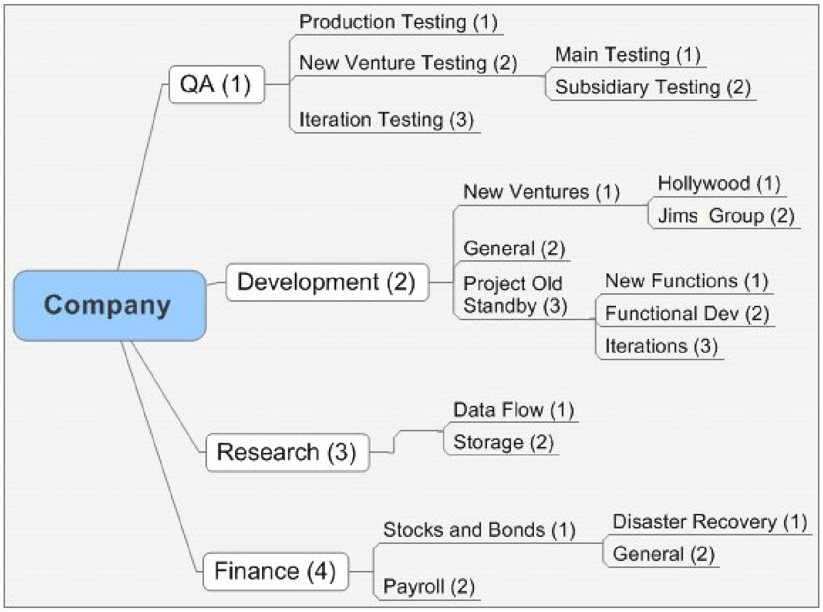


Keep in mind that you want to make sure that your business structure makes a distinction that parallels how you want to manage and distribute your resources. You may want to break out special projects that need dedicated or specialized hosts.

Note:

Your business structure and its hierarchy should reflect long-term business goals because it can be complicated to modify the tree later on.

b) Prioritize all your lower level business areas relative to other leaf consumers from the same branch (siblings).



You now have the basic planning information that you are going to use to create your consumer tree.

How your business structure maps to your tree

The business structure you mapped out above becomes the template to create a consumer tree.

The top-level business processes become top-level consumers.

The lowest area of business becomes a leaf consumer and this is the consumer location where you register such things as services or other application managers.

About the consumer tree

The consumer tree organizes consumers into a structure that makes it easy to apply resource plans.

Overview

The consumer tree is closely related to the resource plan. The plan cannot be defined without the tree.

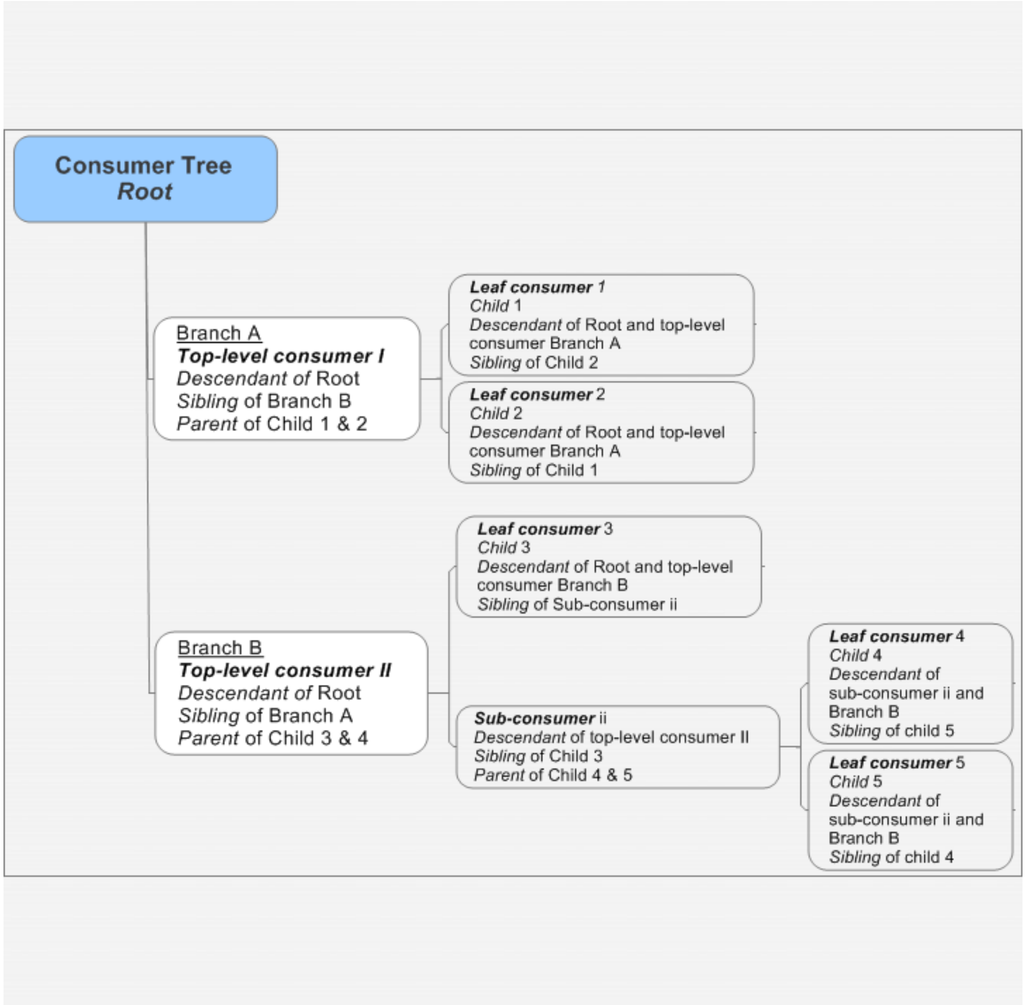
The tree only defines organizational relationships among consumers, while the plan defines resource allocation.

The choice of consumers and their hierarchy should reflect long-term business goals because it can be complicated to modify the tree. To make the system adjust to short-term business changes, you can modify the users associated with a consumer, or the resource plans defined in the plan.

Component	Description
Tree	The resource distribution tree identifies consumers of the cluster resources, and organizes them into a manageable structure.
Plan (Resource plan)	The resource plan describes the relationship between the consumer tree and resource groups, and defines plans for how cluster resources are to be shared among consumers.
Consumers	A consumer in the tree represents any entity that can demand resources from the cluster. A consumer might be a business service, a business process that is a complex collection of business services, an individual user, or an entire line of business. The consumers ManagementServices, SampleApplications, ClusterServices, SymExec, and SymTesting along with their sub-consumers, are installed by default. • ManagementServices has two sub-consumers, SymphonyManagementServices and EGOManagementServices, which run important system services on management hosts in the cluster. Services include derbydb, plc, purger, RS, ServiceDirector, WEBGUI, and WebServiceGateway. ManagementServices is configured to use the ManagementHosts resource group. Do not modify or delete this consumer. • SampleApplications has two sub-consumers, EclipseSamples and SOASamples. The SampleApplications consumer and its sub-consumers can be modified or deleted (although you may want to use the provided samples to begin using Symphony right away).
	• ClusterServices is configured to use the InternalResourceGroup resource group. It has two sub-consumers, SymphonyClusterServices and EGOClusterServices, which runs essential system services on every host in the cluster. Do not modify or delete ClusterServices, or use it to run workload units.
	• SymTesting has one sub-consumer, Symping51, which runs the Symphony configuration testing tool, Symping.
	• SymExec has one sub-consumer, SymExec51, which lets you submit remote commands to an application.
Multi-level tree	Company projects are generally structured with multiple layers and components. For example, a project belongs to a department, a department to a business unit and so on. A multi-level consumer tree allows you to configure consumers in a hierarchical fashion to match your business structure. **Note:** As a best practice, restrict the number of tree levels to four.
Tree root	The root of the tree represents the entire cluster and all resources in it. Resources from the root are distributed through the tree to consumers.
Top-level consumers	The consumers attached directly to the root are called top-level consumers. The top-level consumer is the head of a consumer branch.

Component	Description
Leaf consumers	If a consumer has no descendants, it is called a leaf consumer. Services and applications can only be associated with leaf consumers. Borrow and lend policies are set at this level. **Note:** As a best practice, limit leaf consumers to fewer than 20.
Branches, descendants	If a consumer in the tree has other descendants, thereby creating a branch in a multi-level tree, it is called a branch consumer. Branch consumers exist to redistribute resources down the branch to their descendants. Descendants of a branch consumer may also have descendants, thereby becoming branch consumers themselves. Every branch in the tree ends with a leaf consumer.
Parent	A consumer containing another consumer (a "child"). A parent can contain a child consumer, or be the child of another parent consumer.
Sibling	Two or more consumers sharing the same parent consumer.
Child, sub-consumer	A consumer nested within another consumer (a "parent"). A child (or sub-consumer) of one parent can be the parent to another nested child. A leaf consumer is always a child at the end of the branch.
Consumer administrators	For ease of management, you can create consumer administrators for top-level consumers in a multi-level tree. These users can change the plan for lower-level consumers on their branch (descendants), without requiring cluster administrator permissions. Only a cluster administrator can change the plan for top-level consumers.

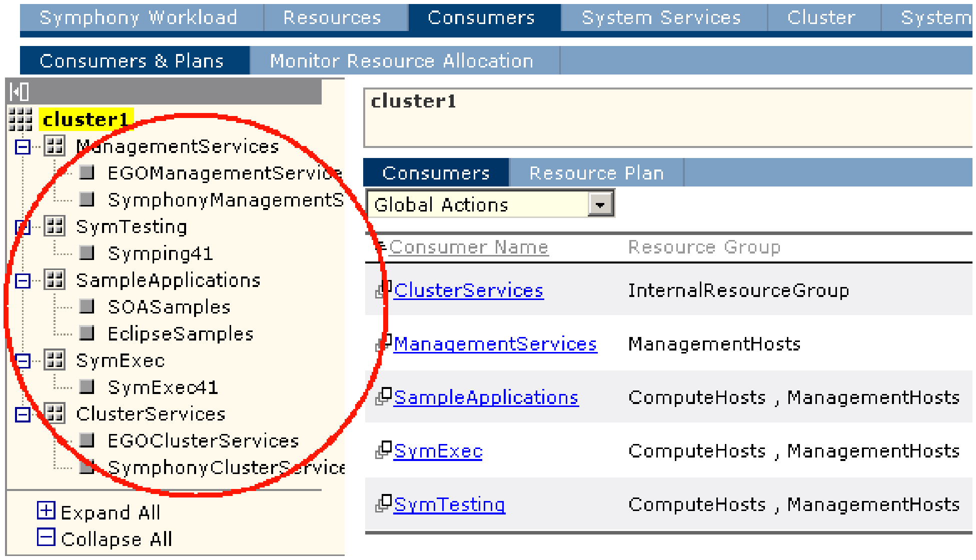
Graphical description



Recognize the default configurations

The default resource components you see and work with in the Platform Management Console let you quickly and easily start using your cluster to run work.

- Consumers:
  - ManagementServices (with nested child consumers SymphonyManagementServices and EGOManagementServices)
  - SampleApplications (with nested child consumers SOASamples and EclipseSamples)
  - ClusterServices (with nested child consumers SymphonyClusterServices and EGOClusterServices)



cluster1
- ManagementServices
  - EGOManagementServic
  - SymphonyManagementS
- SymTesting
  - Symping41
- SampleApplications
  - SOASamples
  - EclipseSamples
- SymExec
  - SymExec41
- ClusterServices
  - EGOClusterServices
  - SymphonyClusterServic

cluster1

Consumer Name	Resource Group
ClusterServices	InternalResourceGroup
ManagementServices	ManagementHosts
SampleApplications	ComputeHosts , ManagementHosts
SymExec	ComputeHosts , ManagementHosts
SymTesting	ComputeHosts , ManagementHosts

Never remove the ManagementServices or ClusterServices branches, or change their consumer names, when building or modifying your tree.

Create your consumers

You have created a diagram mapping out your business structure. You have read the descriptions of the parts of the tree and understand how they interact. You are logged on to the Platform Management Console as a cluster administrator.

It is difficult to reorganize your tree structure once you have created it, so preparation and planning are key. You need to create a consumer for each label on your business structure that you created above. If you have created user accounts and know who the consumer administrator is for each top-level consumer, you can specify those user accounts during this procedure. If not, you can assign consumer administrators later.

1. Click Consumers > Consumers & Plans > Consumers.

   A list of existing top-level consumers displays. By default, you have ManagementServices, SampleApplications, and ClusterServices.

2. Create a consumer for your most important top-level business area.

   Use the plan you have already laid out and create a consumer with the name of that area. From the example above, the first consumer is "QA".

   a) Select Global Actions > Create a Consumer.
   b) Specify the name for your most important consumer.

      The most important top-level consumer in the plan above is “QA”.
   c) If you want, specify administrators and users, or assign them later.
   d) Specify an OS user account associated with this consumer.
   e) Leave the default resource groups selected.

      If you have already created more resource groups or replaced ComputeHosts with others, select as many as you would like available throughout this branch of the tree. If you do not select ManagementHosts and InternalResourceGroup resource group plus at least one other group, you have insufficient resources to run your applications on this entire branch.

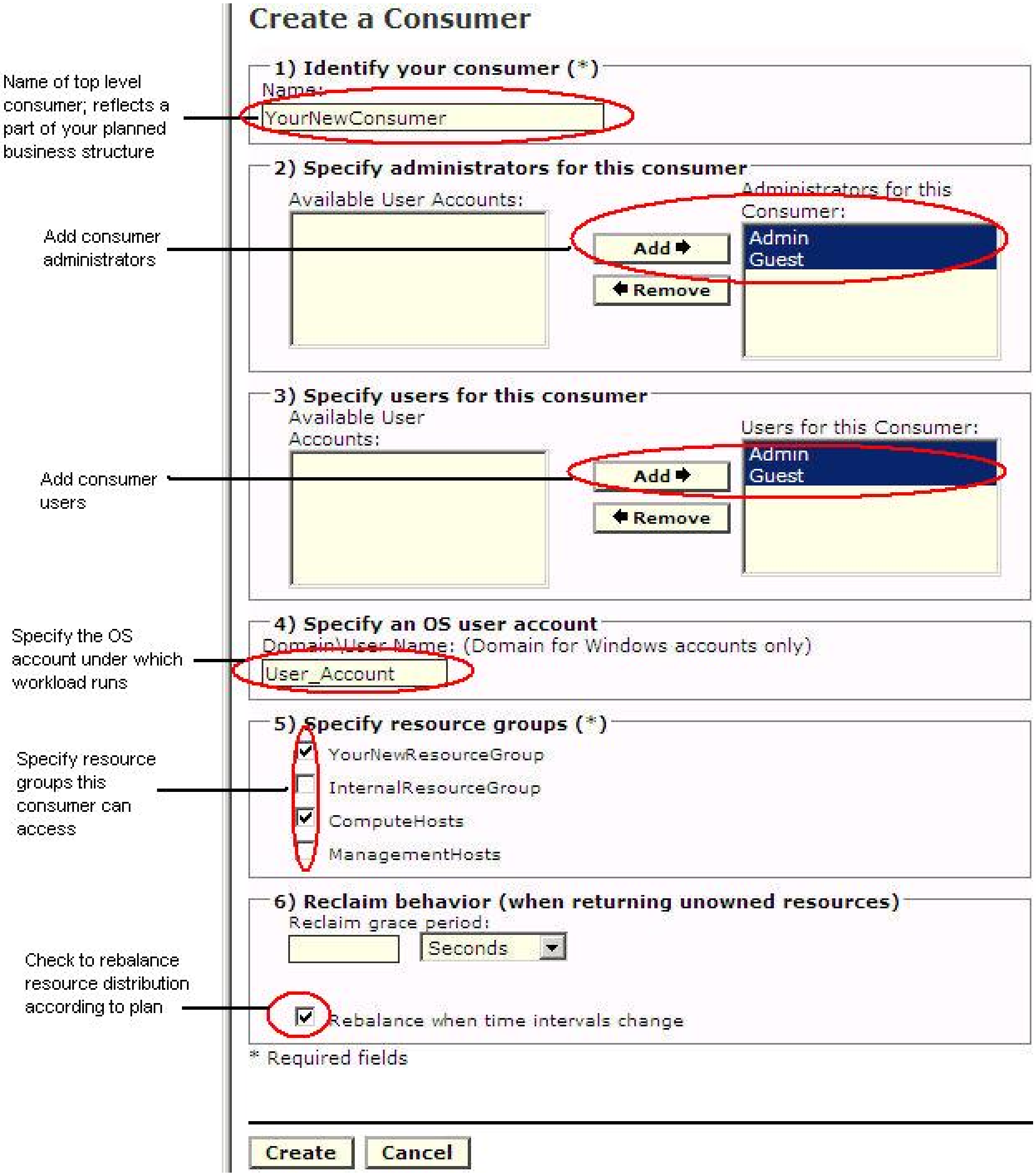
f) Leave the Reclaim behavior section blank.

   Reclaim behavior is an advanced feature and must be coordinated with the resource plan settings.

g) For all consumers (except for those in the ManagementHosts and InternalResourceGroup resource groups), ensure the box Rebalance when resource plan changes or time interval changes is checked within individual Consumer Properties dialog boxes.

   This ensures that when your resource plan changes according to set time intervals, that originally configured share ratios, allocations, and lend/borrow policies are reapplied and enforced across all consumer branches in the consumer tree.

h) Click Create.



Name of top level consumer; reflects a part of your planned business structure

Add consumer administrators

Add consumer users

Specify the OS account under which workload runs

Specify resource groups this consumer can access

Check to rebalance resource distribution according to plan

Create a Consumer

1) Identify your consumer (*)

Name: YourNewConsumer

2) Specify administrators for this consumer

Administrators for this Consumer: Admin, Guest

3) Specify users for this consumer

Users for this Consumer: Admin, Guest

4) Specify an OS user account

Domain\User Name: (Domain for Windows accounts only) User_Account

5) Specify resource groups (*)

☑ YourNewResourceGroup
☐ InternalResourceGroup
☑ ComputeHosts
☐ ManagementHosts

6) Reclaim behavior (when returning unowned resources)

Reclaim grace period: Seconds

☑ Rebalance when time intervals change

* Required fields

3. In the order of the priorities you have already established, create the rest of your top-level consumers.

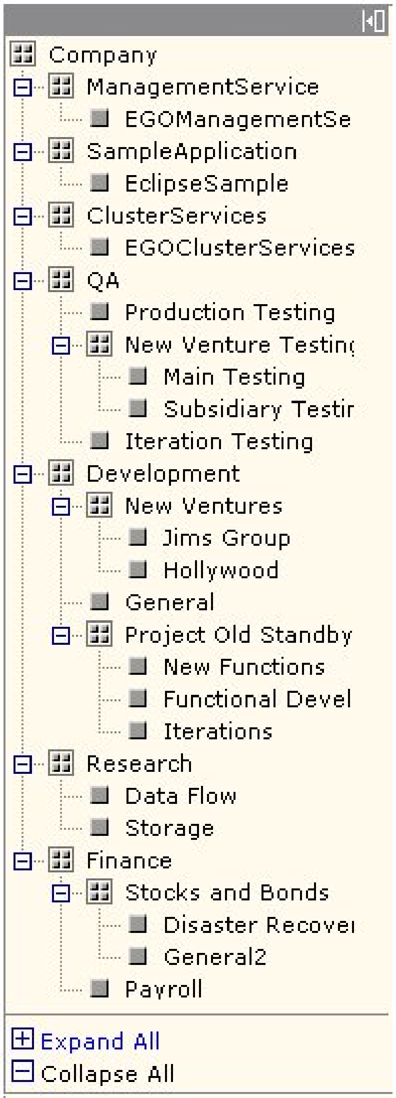
   For example, following the plan above, create Development, Research, and then Finance.
4. Once you have all your top-level consumers in the order that you want, in the Platform Management Console use the tree to navigate to your first top-level consumer (for example, “QA”).
5. Create all the sub-consumers for the most important top-level consumer (for example, “QA”) in the order that you have prioritized them.

6. Navigate to each sub-consumer and create required leaf consumers.
7. Repeat this process until each branch of the tree is complete and matches the plan you made.

The final consumer tree follows the same structure as the business structure.

Tip:

Do not name any two consumers with the same name. In the example below, the number “2” was added to the second consumer named “General” to distinguish them.



The next steps

Now that you have a consumer tree that mirrors your business structure and is ordered according to the most important areas, you are prepared to go register applications to leaf consumers and customize your resource plan.

You are also ready to add user accounts and assign administrators to consumers.

CHAPTER

8

Resource Distribution Plans

Resource plan: an overview

The resource plan defines how cluster resources are allocated among consumers. The plan takes into account the differences between consumers and their needs, resource properties, and various other policies concerning consumer rank and the allocation of resources.

Dynamic allocation overview

For dynamically allocated resources, client demand is fundamental to the allocation. If there is competition from other consumers, or limits configured for the consumer, the allocation could be less than what is needed. This is considered a shortfall.

The priority for allocation is to satisfy each consumer's reserved ownership, then allocate remaining resources to consumers that have demand. By default, resources are systematically allocated in the following order:

1. EGO always allocates owned resources first to consumers according to the resource plan.
2. Then, when there are no owned resources left to borrow from consumers who are willing to lend them, EGO allocates unowned resources from the share pool to consumers with unsatisfied demand.
   - Share pool resources first come from the “family” pool (any unowned resources within a particular branch in the consumer tree).
   - Once the family pool is exhausted, then EGO distributes resources from other branches in the consumer tree, eventually moving up the tree to distribute any unowned resources from the cluster level.
   - Share pool resources are distributed to competing leaf consumers according to their allowed share ratio. If the share ratio for two competing consumers is 1:1, resources are allocated evenly. If the share ratio is 1:2, then the second competing consumer gets twice as many available resources from the share pool.
3. Finally, when there is demand, consumers lend and borrow resources from each other.

Note:

If a consumer has previously lent out resources to another consumer, the lending consumer recalls the owned resources necessary to meet its current unsatisfied demands once other allocation options have been exhausted, regardless of the borrowing consumer’s priority. This default behavior can be changed so that owned resources are instead recalled first before borrowing from other consumers.

Resource update cycle

By default, EGO updates the resource information used for scheduling every 60 seconds. The frequency of the update cycle is determined by EGO_RESOURCE_UPDATE_INTERVAL in `ego.conf`.

The update cycle determines how quickly EGO detects a new resource or unavailable resource in the cluster. It also determines how often EGO detects changes in load indices that might be used to allocate resources to jobs.

Customizing the resource plan

A general order of events for customizing a resource plan might be as follows:

1. If resource groups are necessary, create them first.
2. Create consumers and edit the plan.

3. Customize the resource plan for each consumer.
4. Create a resource plan for each new resource group you add.

The plan can be modified by the cluster administrator and the consumer administrator. The consumer administrator has restricted permissions.

Out-of-box resource plan

EGO installs with two top-level consumers: ManagementServices and SampleApplications. Do not remove the ManagementServices branch: it contains required system services.

The default resource plan can be modified as required (except for removing the ManagementServices branch). You need to create a consumer for each new application.

Important:

If you want to keep the default plan intact (to reset back to it at a later time), be sure to export and save it locally before making any changes. Import it when required.

Reserving management host slot to run services

By default, there are 12 slots per management host. Although this number is configurable, be aware that each service requires at least 1 slot to run. For example, if you configure the EGO Services leaf consumer with 0 owned slots, the cluster does not run. Note that slot ownership must filter down through the consumer tree until a service inherits a slot directly from a parent. To avoid service interruption and to protect a service's owned resource, ensure the following:

- Disable borrowing and lending on the service slot.
- Set the share ratio to 1.
- Set the priority uniformly across all consumer parent/child levels for each service.

Time-based resource planning

Time-based resource distribution

The resource plans can change according to time of day.

Time-based configuration in the resource plan allows the resource distribution for a consumer to change according to the time of day.

Time interval boundaries are at the hour point, so the minimum time interval is 60 minutes. The daily pattern repeats itself every 24 hours.

The simplest configuration is to divide the 24-hour period into 2 intervals such as daytime and nighttime, but you can specify multiple time intervals in the plan, with different resource distribution during every interval.

Example

For example, a consumer requires an ownership of 20 slots from 8:00 a.m. to midnight and only 10 slots from midnight to 8:00 a.m. Define two time intervals: one with ownership=20, one with ownership=10.

If the consumer wants the option to borrow resources only from 8:00 a.m. to 4:00 p.m., three time intervals are required:

- midnight-8:00 a.m. (ownership=10, borrowing disabled)
- 8:00 a.m.- 4:00 p.m. (ownership=20, borrowing enabled)
- 4:00 p.m.- midnight (ownership=20, borrowing disabled)

Resource distribution changes

When a new time interval begins, with one or more consumers having new values for ownership or borrowing configuration, EGO responds to the change in plan immediately.

The change in plan can trigger reclaim of resources. Examples of how plan changes can affect consumers include the following:

Increased ownership	Decreased ownership	Decreased share pool
A consumer whose ownership increases when a new time interval begins may find themselves suddenly under-allocated. If there are workload units pending, EGO distributes more resources immediately.	A consumer whose ownership decreases when a new time interval begins could have workload units interrupted as resources get redistributed or reclaimed by other consumers. When a new time interval begins, resources are immediately reallocated with changes in ownership, regardless of demand considerations. After the new time interval's ownership allocations are made, resources may be reallocated back to consumers with demand through configured borrowing or sharing policies.	In cases where the resource distribution model changes from full share (with no ownership) to a hybrid model (with both sharing and resource ownership), a decrease in share pool resources between plans or given time intervals could affect configured consumer allocations. If there are not enough unallocated resources in the share pool at the moment a new resource plan or time interval takes effect, then consumers are not allocated the planned number of owned resources. Allocated resources from the share pool must first be released back to the share pool by the clients that are using them before they can be reallocated as owned resources to a consumer.
	A consumer whose ownership decreases when a new time interval begins could continue running workload units without interruption in this situation: if borrowing is enabled, the resource status can change from owned to borrowed.	

Understanding the default plan

About the default resource distribution plan

The Symphony installation comes with a default resource distribution plan. Consider the default resource distribution plan to decide whether you need to make any changes.

Note:

If you installed Symphony DE, this topic does not apply.

To allocate the resources you want when you want them, work with your resource distribution plan to distribute the resources for each consumer and each resource group.

Simple by default

A simple 24-hour default resource distribution plan is available to adapt for your own purposes. However, because the default resource distribution plan uses only one time interval (0:00 to 24:00), all current allocations persist until you modify the plan.

Ownership, lending, and borrowing for default consumers

For SymTesting, SOASamples, and EclipseSamples consumers, ownership of compute host slots is not defined, but borrowing is enabled with no limit. This ensures that if compute host resources are available, any of these consumers can make use of them if it requires them.

The ManagementServices consumer does not use compute host slots.

Management host slots are distributed to each consumer through ownership to ensure that system services have enough resources to start.

Default consumers and sub-consumers

The following consumers and sub-consumers are installed by default:

- ManagementServices
  - EGOManagementServices
  - SymphonyManagementServices
- SymTesting
  - Symping51
- SampleApplications
  - SOASamples
  - EclipseSamples
- SymExec
  - SymExec51
- ClusterServices
  - EGOClusterServices
  - SymphonyClusterServices

Symping51 gives you a pre-deployed `symping` application that lets you test your cluster.

SOASamples and EclipseSamples are blank consumers for your own use. For example, you can deploy the sample applications that come with Symphony DE.

Default resource groups

The default plan includes the following resource groups:

- ComputeHosts: All the hosts in your cluster that are not management hosts or being used for internal processes. These are the hosts that should run work.
- InternalResourceGroup: A resource group to run internal processes.
- ManagementHosts: A resource group that only contains those hosts that have the management host package installed.

Why modify the resource plan?

The default resource plan is meant to be immediately useful during the initial stages of installing and testing your cluster.

When it is time to create your own consumers and put the application into production, you will probably modify the default resource plan to take full advantage of all the grid resources and sharing policies that are available.

Advanced features

The default resource distribution plan is meant to be as simple as possible so you can add complexity as you come to understand your grid needs.

The default configuration is intended to make it easy to run the test applications. We do not expect you to keep this configuration. When you create your own applications and define your own consumers, configuring the resource plan and tailoring it to your needs throughout the day can greatly affect your productivity and efficiency.

Tips

- The Management Console stores only your active resource distribution plan. Therefore, to make changes, you can either import a new resource plan or directly edit the resource plan from the Management Console.
- If you are going to change the default resource distribution plan, export that plan before you make changes so you can reset to your previous (if required) by importing it again.
- When editing the resource distribution plan in the Management Console, remember that any changes you make and apply to the active plan are implemented immediately. Do not click Apply to your resource plan changes if you do not wish the changes to take effect immediately. Export the plan instead.

Default resource allocation policy

When a consumer experiences demand, EGO considers resources from each of these five areas, and systematically allocates them (by default) in this order according to the configured resource plan:

1. Idle resources already owned by the consumer
2. Idle, unowned resources from the share pool
    1. Unowned resources within a particular branch in the consumer tree
    2. Unowned resources from other branches in the consumer tree, eventually moving up the tree to distribute any unowned resources from the cluster level
3. Idle resources owned by other consumers that are configured for lending (borrowed resources)
4. Resources owned by the consumer but currently lent-out to other consumers (reclaimed resources to owner)
5. Unowned resources from the share-pool but currently in use by consumers with a smaller share-ratio (reclaimed resources to share-pool)

You can change the default resource allocation policy so that owned resources get reclaimed by consumers before they are borrowed or allocated from elsewhere . You can also adjust the allocation policy so that resources are never reclaimed by the share pool, but are only returned when the borrowing client releases them.

The following table illustrates the revised order of resource allocation depending on configured policy changes.

Default allocation order	Changed policy: Reclaim before borrow	Changed policy: No reclaim on share pool
Idle resources owned by consumer	Idle resources owned by consumer	Idle resources owned by consumer
Idle, unowned resources from share pool	Idle, unowned resources from share pool	Idle, unowned resources from share pool
Idle resources borrowed from other consumers	Resources reclaimed by consumer	Idle resources borrowed from other consumers
Resources reclaimed by consumer	Resources reclaimed by share pool from over-allocated consumers and then re-allocated to deserving consumers with a higher-share ratio	Resources reclaimed by consumer
Resources reclaimed by share pool from over-allocated consumers and then re-allocated to deserving consumers with a higher-share ratio	Idle resources borrowed from other consumers	

Owning and borrowing resources

Ownership

When consumers “own” resources, they are guaranteed a minimum allocation of resources, regardless of competition from other consumers. Ownership is expressed as a numeric quantity.

Ownership is optional. A consumer may not own any resources yet still use cluster resources allocated to them through borrowing. Consumers can choose to lend idle resources.

Overallocating at the cluster level

It is possible to allocate to the cluster (at the top level only in your resource plan) more slots than are currently listed as available. Only do so if resources are, for some reason, not available when you modify your resource plan, but would usually be part of the cluster and will be available when the time interval you are overallocating for occurs.

Attention:

Only overallocate ownership at the cluster level if you know that resources will be available during the time interval you are setting it for.

Lending resources

Lending is optional. You can enable lending only for leaf consumers who own resources (there are no lend settings available for non-leaf consumers in the resource plan). During periods of low demand, a consumer's resources can be lent to other consumers who have an unsatisfied demand. This kind of resource lending/borrowing relationship between consumers improves the efficiency of the cluster. Without lending, owned resources cannot be shared with other consumers and idle resources are wasted.

Owned resources that are not being used and that have lending enabled get allocated to consumers who have an unsatisfied demand. Qualifying resources are lent in the order of configured consumer rank. For example, in the case where a consumer has resources available to lend, and there are competing consumers with unsatisfied demand, this is what would happen:

- First, the borrowing consumer with the highest assigned consumer rank is allocated as many resources as are available until its demand is satisfied or until its configured borrowing limit is reached.
- Then, any surplus resources are assigned to the competing consumer with the next highest consumer rank.
- The allocation continues down the line of consumer rank until all qualifying resources are allocated or all consumer demands are satisfied.

Lending can occur between consumer branches in the consumer tree, and is not restricted to leaf consumers from the same consumer branch. However, through advanced refinement of the resource plan, leaf consumers can be configured to only lend to and borrow from their siblings.

EGO reclaims resources from a borrowing consumer and returns them to the lending consumer as soon as the lending consumer has an unsatisfied demand. Although ownership of resources guarantees access to them at any time, preconfigured reclaim grace periods may delay the recovery of lent resources. When a cluster or consumer administrator sets the reclaim grace period for a consumer, they should consider the length of a typical workload unit potentially run by a borrowing consumer, along with the urgency of workload units that need to be done by a lending consumer that must reclaim its resources.

Reserving resources by setting a lending limit

A consumer has the flexibility to enable lending on all of its owned resources, or on only a few; those resources without lending enabled are reserved solely for use by the leaf consumer that owns them. The reserved resources do not qualify for lending and are never lent out, even if unused. The lending limit is expressed as a numeric quantity.

Borrowing and sharing

Borrowing refers to the temporary allocation of owned resources from a lending consumer or the share pool to a consumer with an unsatisfied demand.

Sharing refers to the temporary allocation of unowned resources from a “share pool” to a consumer with an unsatisfied demand.

Any client can make use of unused, owned resources that are enabled for lending. The only unused resources that cannot be borrowed are those that are reserved for use solely by a resource owner (that is, resources belonging to a consumer who has not enabled lending).

Borrowing is optional. If borrowing is disabled, the allocation to a leaf consumer never exceeds the configured ownership. Therefore, if borrowing is disabled for all consumers, any unused resources (owned by other consumers) are wasted.

Borrowing resources is on a first-come first-served basis. For example, one leaf consumer can borrow all the available resources in the cluster by being the first to request them. Once all available resources are allocated, other leaf consumers that want to borrow must then wait for a resource to be released.

Borrowing can be enabled for leaf consumers only.

Sharing resources between leaf consumers from the same consumer branch

In cases where leaf consumers from the same consumer branch are competing to borrow resources from the share pool, the share ratio determines the minimum number of resources to allocate to each of them.

The share ratio is configurable. A valid entry for a share ratio is a positive, whole number. Share ratios work in this way:

- By default, all consumers have a share ratio of 1, meaning they share equally.
- A share ratio of 0 (zero) means that a consumer cannot borrow at all from the share pool.
- A leaf consumer with a ratio of 2 can borrow twice as many resources as a competing sibling with a ratio of 1, and half as much as a competing sibling with a ratio of 4.

Other examples of share ratios between competing leaf consumers (siblings):

- Scenario: Two competing leaf consumers (siblings) with equal allocations

A ratio of 1:1 means that both siblings receive 1/2 of the available resources from the parent.

- Scenario: Two competing leaf consumers (siblings) with unequal allocations

A ratio of 1:2 means that one sibling receives 1/3 of the available resources from the parent while the other sibling receives 2/3 of it.

- Scenario: Ten competing leaf consumers (siblings) with equal allocations

A ratio of 1 each means each sibling receives equal resources (1/10th of the parent’s available resources).

Note:

Resource allocation to competing leaf consumers depends in turn on branch ownership and share ratios. Therefore, for consumers on different branches of the consumer tree, an identical share ratio does not imply an identical allocation of resources.

In addition to setting share ratios, the cluster administrator may set maximum shares for each consumer. A maximum share value is specified as an absolute numerical count of resources.

Share ratio enforcement throughout the consumer tree

By default, planned share ratios are enforced at the leaf level. This means that share policies guarantee that each application (registered at a leaf level) receives its planned or “deserved” number of resources when demand is demonstrated. If an application does not have sufficient demand to warrant receiving all its deserved resources, the unused resources are distributed to all consumer branches and filtered down to leaf consumers as per their relative share ratios.

You can change the default behavior to enforce share ratios at the parent level. Doing this forces EGO to distribute unused resources to sibling leaf consumers (within a single line of business) that exhibit demand first before it distributes them throughout the rest of the consumer tree (to other lines of business). This allows a line of business to share resources between its own registered applications before sharing with other lines of business.

Resource allocation according to consumer rank and borrowing preference

Once a consumer branch’s share pool of resources is exhausted, then EGO allocates resources from other branches in the consumer tree, eventually moving up the tree to allocate any unowned resources from the cluster level.

Leaf consumers borrow resources from other consumer branches according to the following policies:

1. Consumer ranking: Leaf consumers from the same consumer branch with the highest priority setting have the first opportunity to borrow.

   Note:

   The cluster administrator can set a maximum number of resources that can be borrowed by each consumer.

2. Borrowing preference order: In cases where resources may be borrowed from multiple sources, lenders are ordered by “borrowing preference”. A borrower’s demands are first satisfied by borrowing from the lender for which he has the highest borrowing preference.

Limited borrowing

By default, a consumer with unsatisfied demand can potentially borrow all qualifying resources. However, you can choose to limit the number of borrowed resources allocated to a specific consumer. The borrowing limit is expressed as a numeric quantity.

In cases where a consumer owns resources and also borrows additional resources, the specified maximum allocation includes both the borrowed and owned resources.

Resource reclaim

A consumer does not retain guaranteed use of borrowed resources. Borrowed resources get returned to their owners in two situations:

- When the borrowing consumer and its client releases them
- When owners reclaim their resources to meet their own unsatisfied demand

Resource reclaim is influenced by the grace period set by cluster or consumer administrators and the configured consumers rank.

Note:

EGO may not always return the exact resource that was originally lent. In cases where a high priority workload unit may be running on a lent resource, an analogous resource may be returned instead to the original lending consumer. This behavior is dependent upon the application manager or consumer (for example, Platform Symphony or an LSF cluster) that may be installed on EGO.

Grace period

Lent resources can be reclaimed by owners experiencing unsatisfied demand even if the client is using them. When a resource is reclaimed, any client workload units running on the resource are interrupted. You can set a grace period, however, to impose a delay before a borrowed resource is returned to its owner. For example:

- If you set the grace period to 10 seconds, any client workload units continue to run on the borrowed resource for 10 seconds after EGO initiates the resource reclaim.
- If you set the grace period to 1, any running client workload units are almost immediately interrupted.

Before setting a grace period, consider the length of a typical workload unit that is run by a borrowing consumer and its clients, and the urgency in which a lending consumer might require its demands be satisfied.

Note:

Leaving the grace period unconfigured or blank uses the default grace period of 0 seconds.

Reclaim according to consumer rank

Resources are reclaimed according to their configured consumer rank.

- Example 1: If a lending consumer has unsatisfied demand and requires that its lent resources be reclaimed, EGO looks to reclaim resources starting with leaf consumers with the lowest consumer rank.
- Example 2: If a lending consumer has a specific resource requirement (for example, the lending consumer needs a Windows slot with a certain amount of available memory), EGO reclaims the first lent resource it finds that matches this requirement. Borrowing leaf consumers with the lowest consumer rank are considered first, followed by leaf consumers with a higher consumer rank.

Change reclaim behavior for owned resources

By default, owned resources are only reclaimed after the lending consumer has attempted to satisfy its unmet demand through all other available means, including by borrowing resources from other lending consumers. You can, however, change this behavior so that owned resources get reclaimed before a consumer attempts to borrow resources from other lending consumers.

Changing the reclaim behavior is useful in cases where a consumer’s owned resources are specially selected to run certain workload units, or in charge-back settings where borrowing from outside a resource group might be more costly.

Change share pool reclaim behavior

By default, share pool resources can be reclaimed. This allows the share pool to reclaim resources from an over-allocated consumer to meet the demands of a competing consumer with a higher share ratio. You can change this behavior so that share pool resources are not reclaimed. Instead, resources get returned to the share pool for further allocation once the borrowing consumer and its client releases them.

Troubleshooting unexpected resource allocation issues

If you find that leaf consumers are not getting enough resources, or that client workload units are not running as expected, check the following:

- Ensure that the entire consumer branch owns adequate resources (that parents own enough resources to meet the demands of their children).
- Check that the priority levels are set appropriately (that they are not all set to “low” or all set to “high”).
- Confirm that the share ratio is appropriate between sibling leaf consumers (that more important leaf consumers are given a higher share ratio than competing siblings).
- Make sure that you enable borrowing and lending.

Share ratio behavior and enforcement

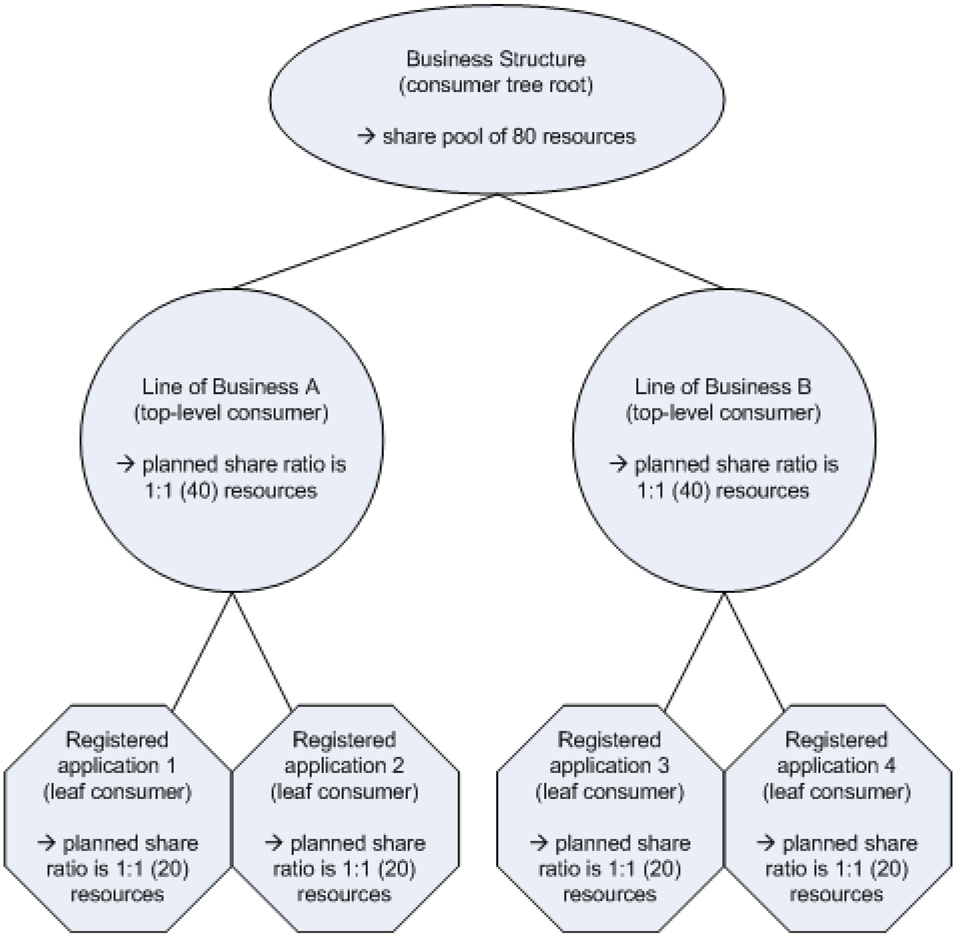
Sharing refers to the temporary allocation of unowned resources from a “share pool” to a consumer with an unsatisfied demand. Behavior of how these resources are distributed to consumers can be configured.

Default behavior—share ratio enforced at leaf level

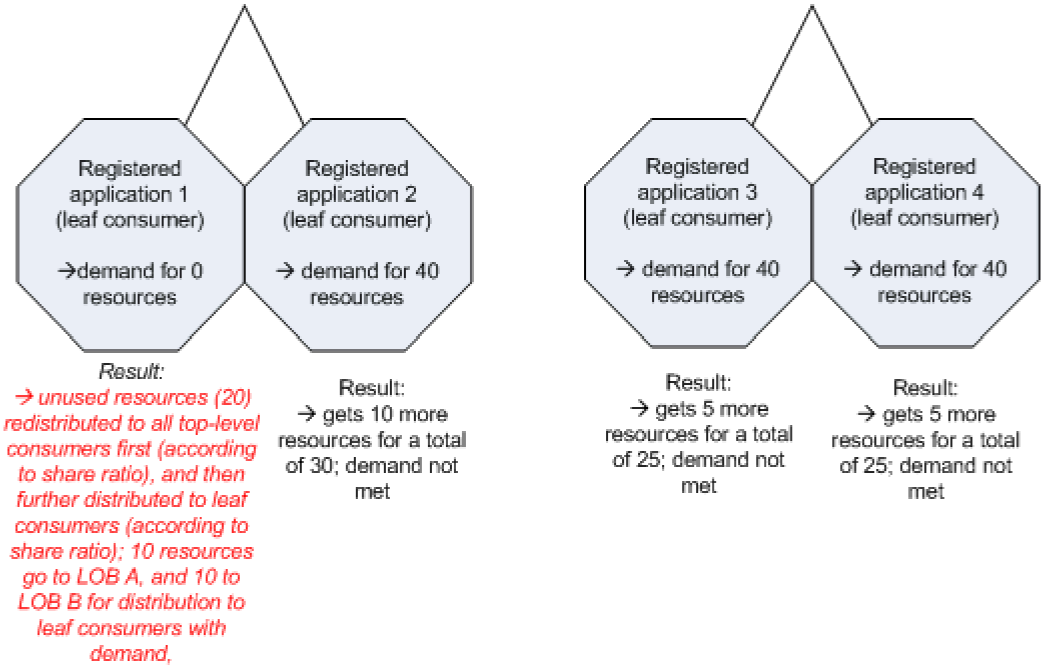
Distribution model

By default, planned share ratios are enforced at the leaf level. This means that existing share policies guarantee that each application (registered at a leaf level) receives its planned or “deserved” number of resources when demand is demonstrated. If an application does not have sufficient demand to warrant receiving all its deserved resources, the unused resources are distributed to all consumer branches and filtered down to leaf consumers as per their relative share ratios.

In the following sample resource distribution model, all applications (registered to leaf consumers) are configured with equal share ratios; each leaf consumer has a 1:1 share of the resources distributed to its parent (top-level consumer from the same branch). Assuming all applications have the same demand, all receive the same number of resources.


Business Structure
(consumer tree root)
→ share pool of 80 resources
Line of Business A
(top-level consumer)
→ planned share ratio is 1:1 (40) resources
Line of Business B
(top-level consumer)
→ planned share ratio is 1:1 (40) resources
Registered application 1
(leaf consumer)
→ planned share ratio is 1:1 (20) resources
Registered application 2
(leaf consumer)
→ planned share ratio is 1:1 (20) resources
Registered application 3
(leaf consumer)
→ planned share ratio is 1:1 (20) resources
Registered application 4
(leaf consumer)
→ planned share ratio is 1:1 (20) resources

Building on the sample resource distribution model above, if one of the applications no longer demonstrates a resource demand, its “deserved” resources are distributed to other leaf consumers in the tree according to their configured share ratios (in this case, the share ratios are equal). This is the default behavior.

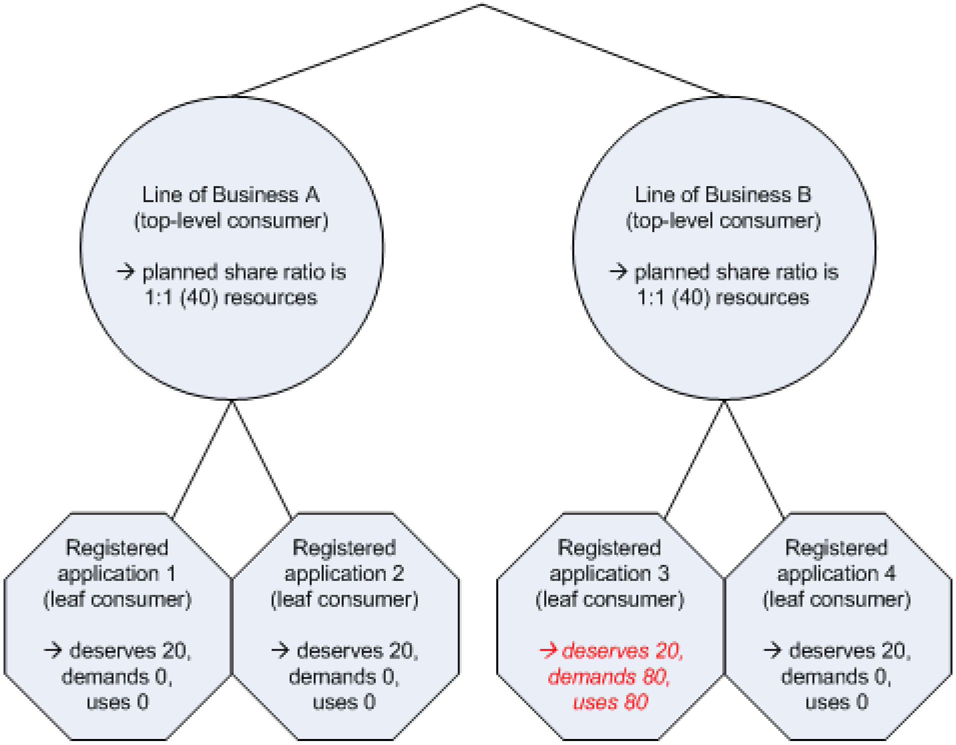

Registered application 1 (leaf consumer)
→demand for 0 resources
Registered application 2 (leaf consumer)
→ demand for 40 resources
Registered application 3 (leaf consumer)
→ demand for 40 resources
Registered application 4 (leaf consumer)
→ demand for 40 resources
Result:
→ unused resources (20) redistributed to all top-level consumers first (according to share ratio), and then further distributed to leaf consumers (according to share ratio); 10 resources go to LOB A, and 10 to LOB B for distribution to leaf consumers with demand,
Result:
→ gets 10 more resources for a total of 30; demand not met
Result:
→ gets 5 more resources for a total of 25; demand not met
Result:
→ gets 5 more resources for a total of 25; demand not met

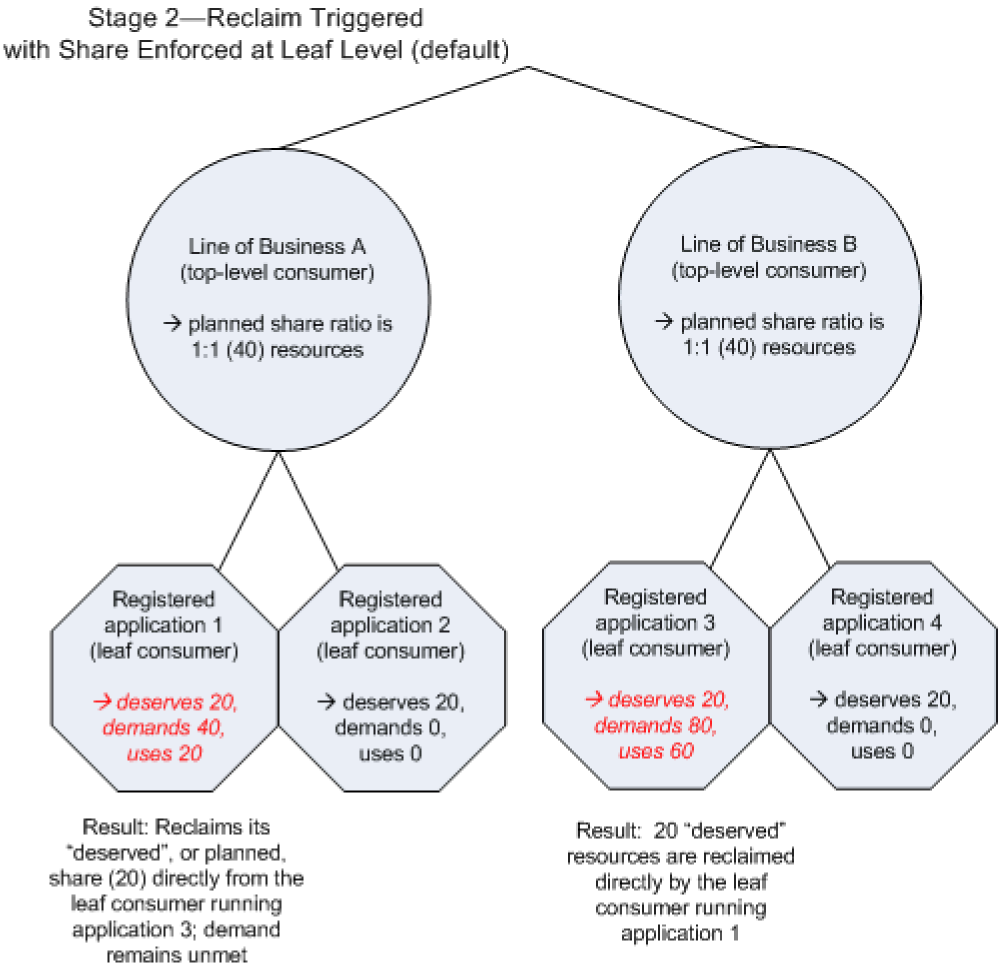
Reclaim behavior

If reclaim is triggered, a leaf consumer takes back its “deserved” number of resources in use by other consumers, up to its planned share. For example, if a leaf resource is experiencing an unmet demand, then it reclaims resources directly from another leaf consumer who is using more than its planned share of resources.

The first leaf consumer reclaims without consideration of the needs or planned share ratio of the second leaf consumer’s branch. The first leaf consumer does not care if the consumer branch it reclaims from falls below the branch’s “deserved” number of resources.

Stage 1—Prior to Reclaim Trigger

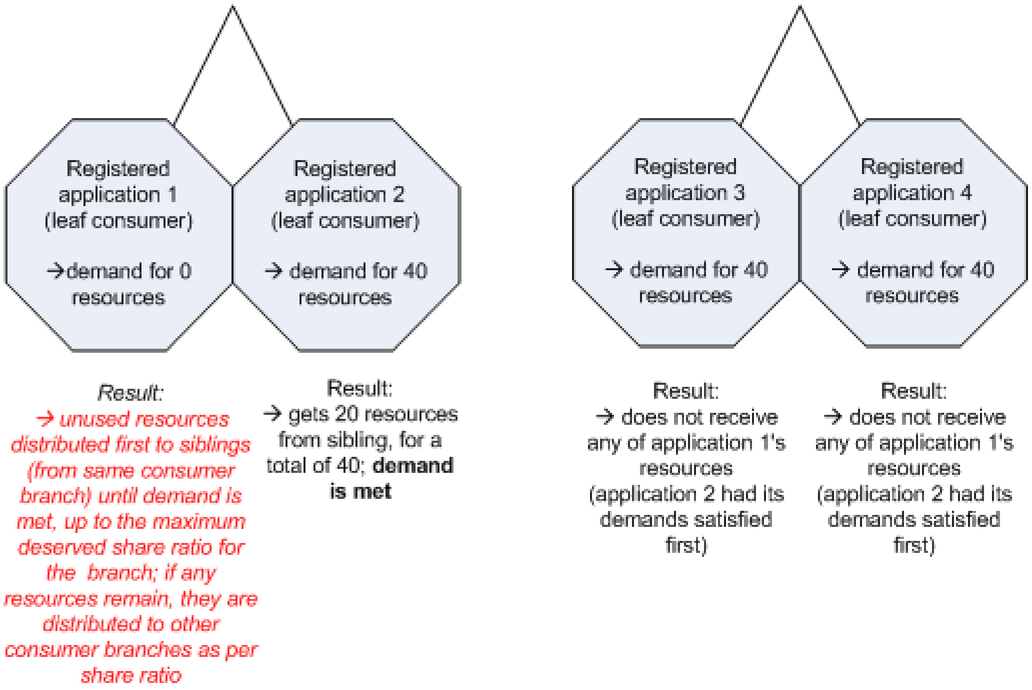




Reconfigured behavior—share ratio enforced at parent level

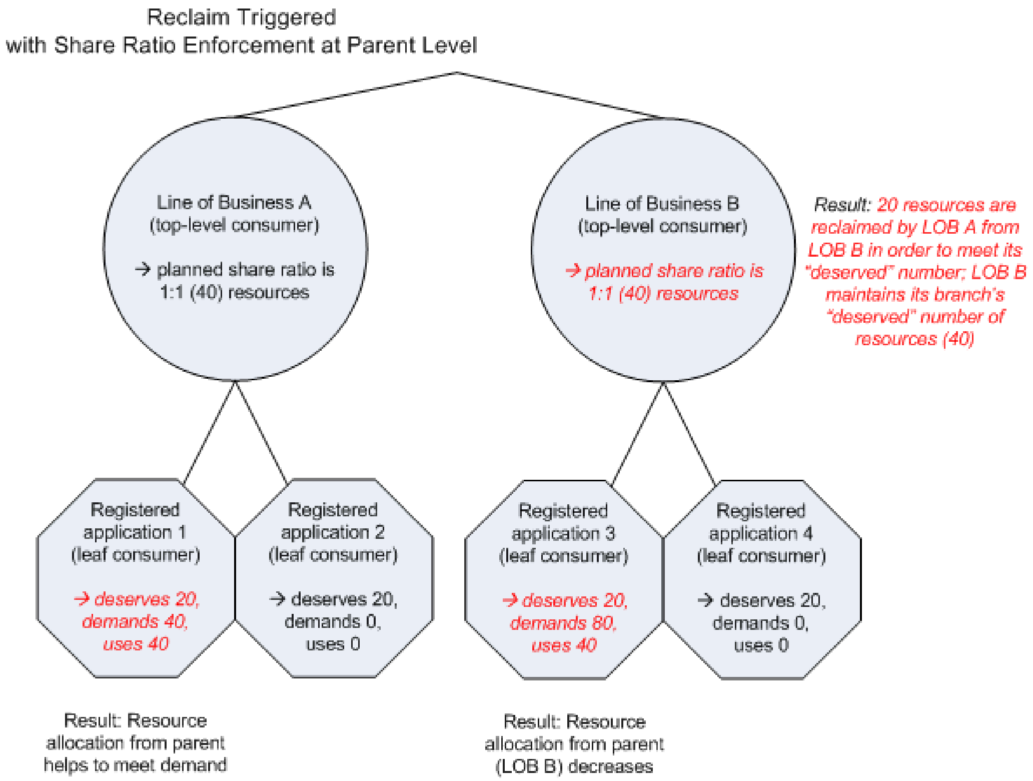
Distribution model

EGO can be configured so that consumers' share ratios are enforced at the parent level instead of at the leaf level. This means that if one of the applications no longer demonstrates a resource demand, its "deserved" resources are distributed first to its siblings (other leaf consumers with registered applications within the same consumer branch) experiencing an unmet demand, and then to other leaf consumers in the tree.


Registered application 1 (leaf consumer)
→demand for 0 resources
Registered application 2 (leaf consumer)
→ demand for 40 resources
Registered application 3 (leaf consumer)
→ demand for 40 resources
Registered application 4 (leaf consumer)
→ demand for 40 resources
Result:
→ unused resources distributed first to siblings (from same consumer branch) until demand is met, up to the maximum deserved share ratio for the branch; if any resources remain, they are distributed to other consumer branches as per share ratio
Result:
→ gets 20 resources from sibling, for a total of 40; demand is met
Result:
→ does not receive any of application 1's resources (application 2 had its demands satisfied first)
Result:
→ does not receive any of application 1's resources (application 2 had its demands satisfied first)

Reclaim behavior

By enforcing share ratios at the parent level, a consumer branch's "deserved" number of resources is considered in cases where reclaim is triggered. Resources are no longer reclaimed directly from a leaf consumer, but are instead reclaimed from the top-level consumer (parent). A parent only releases shared resources if it does not cause unmet demand in its own leaf consumers; if there is demand in its own branch, a parent does not release more resources than its "deserved" share requires.


Reclaim Triggered
with Share Ratio Enforcement at Parent Level
Line of Business A
(top-level consumer)
→ planned share ratio is 1:1 (40) resources
Line of Business B
(top-level consumer)
→ planned share ratio is 1:1 (40) resources
Result: 20 resources are reclaimed by LOB A from LOB B in order to meet its "deserved" number; LOB B maintains its branch's "deserved" number of resources (40)
Registered application 1 (leaf consumer)
→ deserves 20, demands 40, uses 40
Registered application 2 (leaf consumer)
→ deserves 20, demands 0, uses 0
Registered application 3 (leaf consumer)
→ deserves 20, demands 80, uses 40
Registered application 4 (leaf consumer)
→ deserves 20, demands 0, uses 0
Result: Resource allocation from parent helps to meet demand
Result: Resource allocation from parent (LOB B) decreases

Resource sharing: best practices

Overview: Moving from a silo to a distributed sharing model

Novice EGO administrators may implement a “silo” model for initial testing, creating resource groups that remain quite distinct and self-contained, and configuring resource plans to exclude the ability to borrow and lend resources between consumers. By maintaining such a limited model, administrators lose the advantages of EGO’s flexibility to dynamically respond to consumer and client needs and to distribute resources effectively across a cluster. To break down the silo and move to a more distributed sharing model, you must enable lending and borrowing. This can be done by altering the resource plan in a variety of ways, and ranking consumers.

Suggested best practices for resource planning include:

- Configuring resources that are available for sharing
- Configuring surplus resources for lending

Note that the key to effective resource distribution includes more than configuring suitable plans and policies. You must also ensure that there are enough resources distributed to the resource group(s) in order for the lend/borrow plans to make a difference. A transition from a silo to a distributed model of resource distribution requires a combination of effective resource plan configuration and the appropriate distribution of resources to resource groups in the tree.

Configuring resources that are available for sharing

The cluster administrator may set aside a number of resources for all consumers within the consumer tree to share. These unowned resources get distributed throughout the tree according to share entitlements (share ratio).

A best practice to help you better leverage the distribution of resources available for sharing is to interpret the share ratio as a minimum share value. These minimum values are complemented by the maximum share values you must also set. A distribution policy that includes resources available for sharing should always yield actual share values between the minimum and maximum constraints. So long as this remains the primary expectation, the complexities of the actual distribution algorithm do not matter.

Hybrid Scheduling Policy

Scope

Applicability	Details
Operating system	All operating systems supported by Symphony
Symphony	Version 5.1 and higher
Limitations	

Why use the hybrid policy?

The hybrid policy combines elements of both the ownership and sharing policies.

The key features of the hybrid policy are:

1. Resources can be owned and reserved. Reserved resources are excluded from lending and are always dedicated to the exclusive use of one consumer.
2. Sharing (borrowing and lending resources) is enabled among all consumers. There is no need for labor-intensive maintenance of configured of peer-to-peer relationships.
3. When consumers need to borrow more resources than they own, the borrowed resources are distributed in proportion to the amount of resources each consumer owns.

Comparing consumer policies

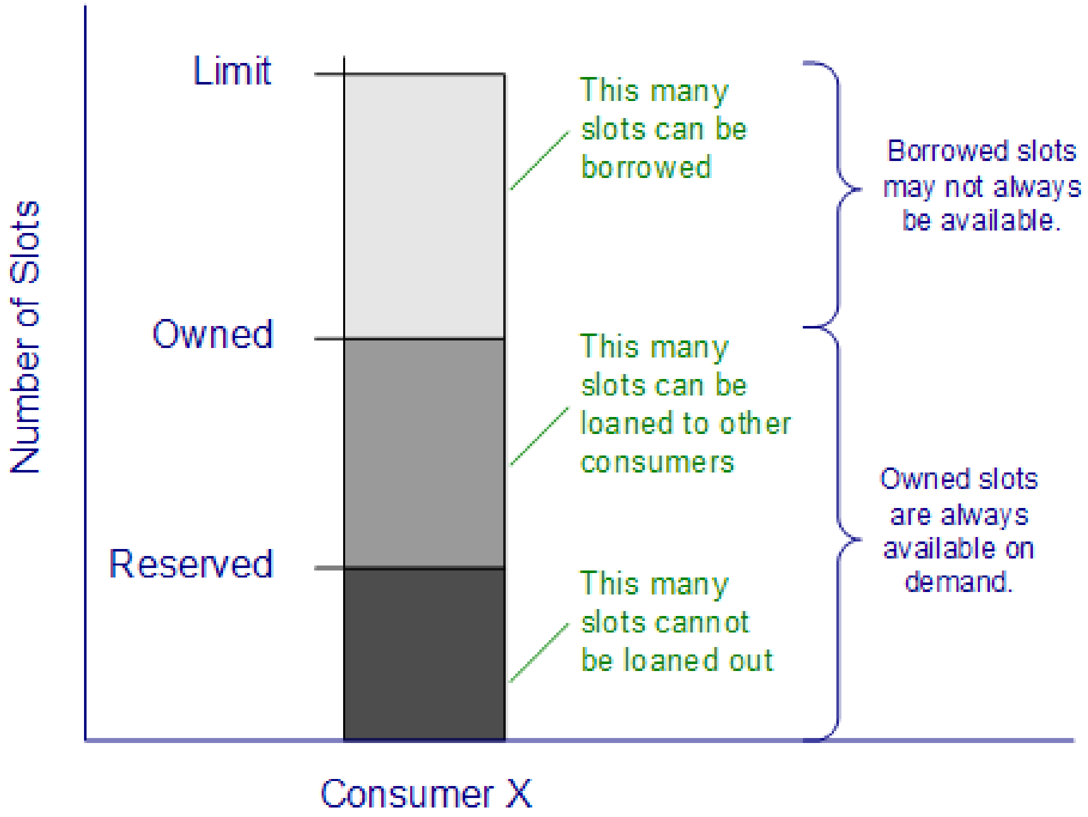
The following table summarizes the attributes of each scheduling policy.

Feature	Hybrid	Ownership	Share ratio
Sharing by default	X		X
Reserve slots from being shared	X	X	
Plan configured by absolute number	X	X	
Sibling first borrowing	X		X
Balance checking	X	X	
Proportional borrowing	X		X
Proportional reclaiming	X		X
Proportional ownership	X		

Basic Concepts of the Hybrid Policy

Basic policy configuration for each consumer that can use the resource group includes the following:

- owned
- reserved
- limit


Limit
Owned
Reserved
Number of Slots
This many slots can be borrowed
This many slots can be loaned to other consumers
This many slots cannot be loaned out
Borrowed slots may not always be available.
Owned slots are always available on demand.
Consumer X

Resources owned

Default = 0. By default, a consumer only borrows slots, it does not own any.

A consumer that owns slots can use them whenever it needs them, even if it has to reclaim them from another consumer. Owning slots in the hybrid policy guarantees that a consumer is entitled to a specific number of slots at any time, even if other consumers also need slots.

Reserved resources

Default = 0. By default a consumer makes all its owned slots available for lending, and does not reserve any. Reservation is only possible if the consumer owns the slots.

A consumer's owned slots are either reserved or available for lending to other consumers (when they are not needed). A consumer can reserve some, all, or none of its owned slots.

- Reserved slots cannot be lent to other consumers. When reserved slots are not needed by the owners, the system reserves enough idle slots to satisfy the potential demand. When an owner needs a reserved slot, it is available immediately.
- If an owned slot is not reserved, it can be used by other consumers and then reclaimed when it is needed.

Note:

Reserving slots can result in the best performance for a consumer, because tasks start without any delay. However, it usually results in the worst efficiency overall, because slots reserved by one consumer can be idle while another consumer has pending tasks.

Resource limit

Default = unlimited. By default, there is no fixed limit to the number of slots a consumer can use.

If you want to prevent a consumer from using too many slots at one time, set a limit. The limit is compared to the total number of slots used by a consumer, it does not matter if the slots are owned, or borrowed, or a combination of both.

Note:

Limiting a consumer's resource usage can actually improve the efficiency across multiple consumers, in cases where consumer workload changes quickly and it would take a long time to redistribute those resources to other consumers. Resource limits prevent the system from taking demand-based slot allocation to extremes. Resource limits help you enforce a more balanced distribution of slots among consumers.

Advanced Concepts of the Hybrid Policy

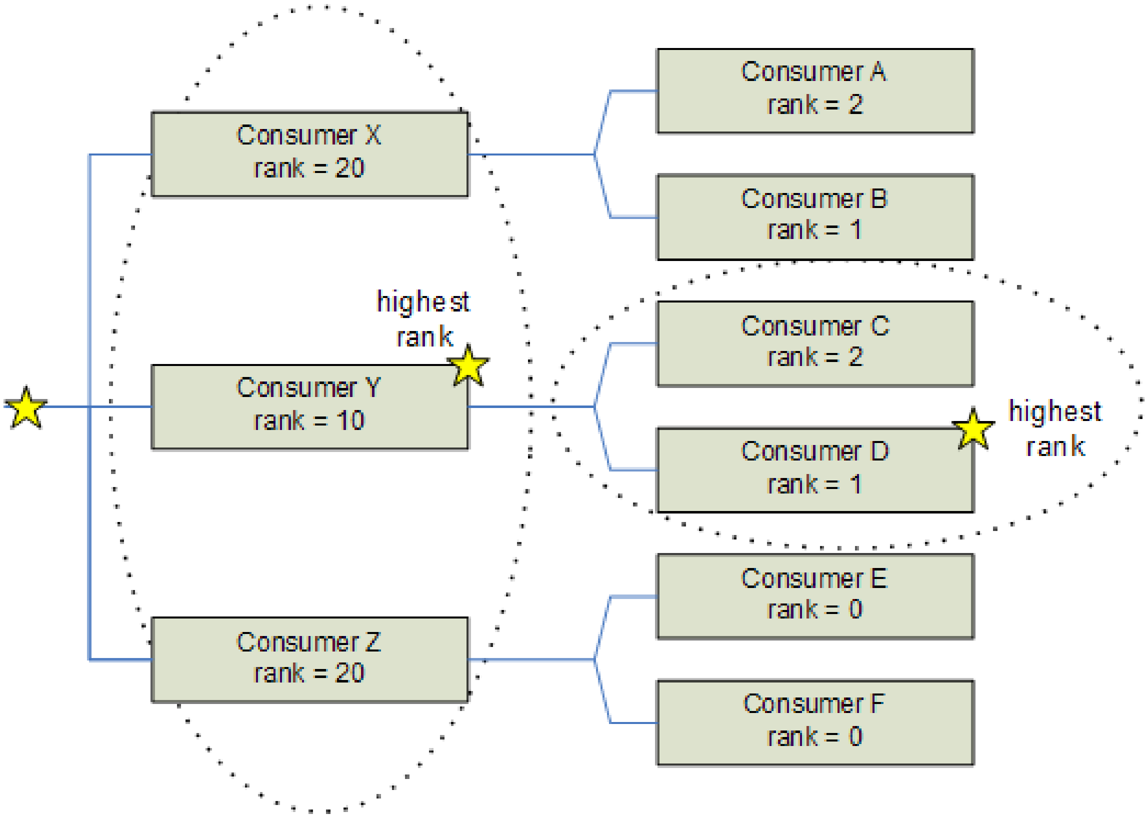
If you want more advanced control over the way slots are shared among consumers, you should understand the following:

- hierarchy
- rank
- quota
- slot distribution
- reclaim

Consumer Hierarchy

To distribute slots to leaf consumers, the system does not necessarily compare all the leaf consumers at once. Instead, the system considers how to divide all the available slots among the root-level consumers. If any consumers have sub-consumers, the slots assigned to each parent are then shared among its child consumers.

Comparing consumers in a hierarchy



Child consumers with the same configuration and the same parent should get the same number of slots. However, child consumers with the same configuration but different parents may get a different number of slots.

In this example, all consumers are busy, and child consumers A,B,C, and D are all configured the same, but parent consumers X and Y are not. The resource group has 60 slots, but X is assigned 10 and Y is assigned 50.

This imbalance affects the child consumers: even though the leaf consumers all have equal demand, A and B must share the 10 slots from X (5 each) while C and D share 50 slots from Y (25 each).

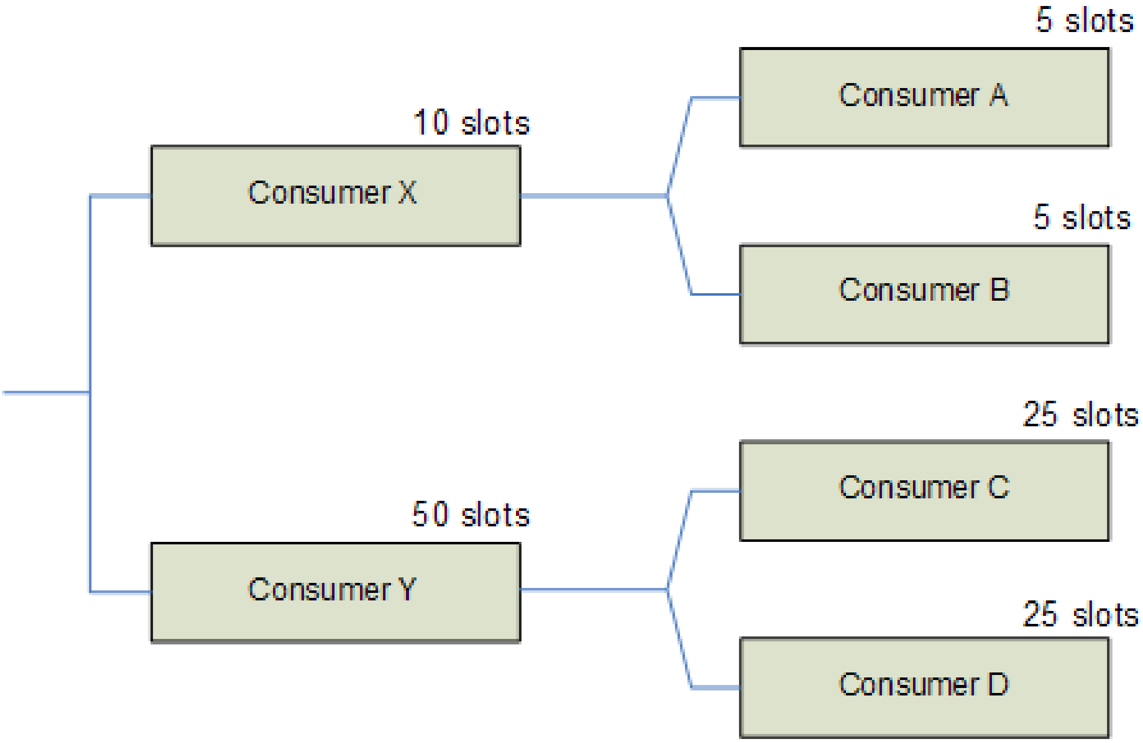
Consumer Rank

Sometimes the system must choose to distribute a slot to a single consumer, but there are multiple consumers that are equally entitled it. This is when consumer rank matters.

If you configure consumer rank, the system distributes slots to the highest-ranked consumers first. Among consumers of equal rank, the system follows the order that they appear in the internal configuration file `ConsumerTrees.xml`; this corresponds to the order that they appear in the complete consumer tree under the Consumers tab.

By default, all consumers have the highest possible rank (0). You can lower a consumer’s rank by specifying a higher numeric value.

Comparing consumer rank



If child consumers have the same parent, it is useful to compare their rank. However, it is not useful to compare the rank of child consumers that have different parents.

In this example, all leaf consumers have equal demand. The parent consumers X,Y, and Z all own the same number of slots, so they are equally entitled to additional resources. However, they have different ranks. If there is only one slot available, it goes to the highest-ranking consumer at the root level (Y) and then passes to its highest-ranking child (D).

The high ranks of consumers E and F do not affect this decision, because their parent consumer had a lower ranking.

Quota Calculations

Every scheduling cycle, Symphony re-evaluates the number of slots from each resource group that each consumer is entitled to use at that time. We call this dynamic number the consumer's "quota".

Even if configuration stays the same, the quota may change because of changes in workload or slot availablity. After calculating the new quotas, the system reallocates slots, so each consumer gets the appropriate amount.

The system tries to set every consumer's quota so it has enough slots to run all its tasks. In a resource group that has plenty of available slots, this may be possible. When there is more competition for resources, the way you configure the policy determines which consumers are the most affected by an overall shortage of resources.

Assuming no reserve or limit, the quota is based on 3 calculations:

1. *How many slots does the consumer need?*
2. *How many slots does the consumer own?*
3. *How many slots can the consumer borrow?*

How many slots does the consumer need?

The number of slots a consumer needs from the resource group is called the "demand".

- For a *leaf* consumer, it corresponds to the number of slots required by applications.
- For a *parent* consumer, the demand is calculated by adding up the demand of each child consumer.

To prevent waste, the quota never exceeds the demand (assuming the consumer has no reserved slots).

How many slots does the consumer own?

The number of slots a consumer owns is proportionally adjusted, and depends on the number of slots in the resource group. The number of slots a consumer owns is based on the value defined in the policy configuration, but the actual value is always adjusted based on the current size of the resource group.

For example, if you configure Consumer X to own 10 slots in a resource group with 100 slots (so X owns 10/100 configured slots), and then you add hosts to the resource group until it actually has 120 slots (without changing the configuration), the quota is calculated assuming that X owns 12 slots, not 10. If some hosts become unavailable, and the resource group has only 90 slots available, the quota at that time is calculated assuming that X owns only 9 slots.

How many slots can the consumer borrow?

The number of slots a consumer is entitled to borrow is proportional to the number of slots it owns. For example, if sibling consumers own 10 and 20 slots respectively, and still need more slots, any additional slots they share will be distributed in a 1:2 ratio.

If sibling consumers own 0 slots, any slots they borrow are shared equally among them.

Calculating the quota with reserve and limit

In more complex cases, reserve and limit affect the quota calculation:

- The maximum quota is the limit, even if the consumer has demand greater than its limit. The value of the limit does not depend on the size of the resource group.
- The minimum quota is the reserve (if a consumer reserves slots, the quota can exceed the demand, and slots can be idle).

  The reserve is not normally adjusted based on the size of the resource group. For example, in a resource group with 100 slots configured, suppose you configure Consumer X to own 10 slots and reserve 6 (so X owns 10/100 configured slots and reserves 6):

  - If the resource group doubles in size, or reduces in size to 80 slots, the number of slots owned is adjusted proportionally, but the quota is still calculated assuming X reserves 6 slots.
  - However, if the number of slots in the resource group is reduced to 50, the number of slots owned by X becomes 5. The consumer cannot reserve more slots than it owns, so the quota is now calculated assuming X reserves 5 slots.

Calculating the quota with rank

Rank changes the quota when the mathematical calculation of the quota results in a fraction (the system must allocate a whole slot, or none at all).

For example, if Consumer A owns 1 slot and Consumer B owns 2, any slots they can borrow are shared in a 1:2 ratio. If there are 10 slots, the system cannot really allocate 3.333 slots to A and 6.666 to B, so the final quota depends on rank:

- If A has the higher rank, its quota of 3.333 is increased to 4, and B's quota becomes 6.
- If B has the higher rank, its quota of 6.666 is increased to 7, and A's quota becomes 3.

Calculating the quota in a hierarchy

If the consumer tree includes sub-consumers, the system does not compare all the leaf consumers at once. Instead, the quota is calculated among the root-level consumers first, then the slots assigned to each parent are shared among its child consumers.

- For a parent consumer, the demand is calculated by adding up the demand of each child consumer.
- Among child consumers, the quota is calculated by dividing up the quota assigned to their parent consumer.

Slot Distribution

The quota is the number of slots a consumer is entitled to have. If the number of slots actually allocated to the consumer is different, the consumer is either over-allocated or under-allocated, and the system needs to reallocate slots.

After calculating the new quotas, the system reallocates slots, so each consumer gets the correct amount.

1. The first step is this process is the distribution of available slots; the system distributes any idle slots to consumers that are under-allocated (compared to their quota).

   When distributing slots, the system allocates the all the available slots to the root-level consumers. If any consumers have sub-consumers, the slots assigned to each parent are then shared among its child consumers.

   The order of distribution matters because the number of idle slots is not usually enough to meet all the quotas. The first consumers to get slots will get the slots that are available immediately, and the last consumers to get slots will get the slots that have to be reclaimed.

   1. At the root level, distribute all the idle slots to under-allocated consumers until they reach their quota, in order of rank. (Among consumers of equal rank, the system follows the order that they appear in the internal configuration file `ConsumerTrees.xml`; this corresponds to the order that they appear in the complete consumer tree under the Consumers tab.)
   2. If the consumer is a parent, the slots are passed down to its child consumers, also in order of rank.
2. The second step in the process is reclaim. If all the idle slots are distributed and some consumers are still under-allocated, the system will reclaim slots.

Reclaim

After calculating new quotas, the system reallocates slots, so each consumer gets the appropriate amount. If the system has no idle slots to distribute, but some consumers are still under-allocated (compared to their quotas), it means at least one other consumer is over-allocated.

To reclaim slots from consumers that are over-allocated, the system terminates and requeues any task that is running on the slot. The idle slot is then allocated to another consumer.

Reclaim is initiated at the end of a scheduling cycle, and if the task releases the slot quickly, the slot becomes available for the system to distribute in the next scheduling cycle. The under-allocated consumer that is waiting for the slot waits for at least one scheduling cycle. The actual time to release the slot depends on system overhead, time to clean up after terminating a task, and the reclaim grace period.

Reclaim grace period

By default, there is no reclaim grace period, and a task is killed immediately to reclaim the slot.

The reclaim grace period is intended to improve overall efficiency of resource usage. When a slot must be reclaimed, and a grace period is defined, the task will keep running until it completes, or until the grace period expires, whichever comes first. If the task fails to complete before the grace period expires, it is terminated and requeued.

- If the task finishes within the time allowed, it releases the slot immediately (instead of waiting for the full grace period).
- If the task is still running when the grace period expires, the task is terminated and requeued.

You configure the grace period separately for each consumer. The over-allocated consumer (the one running the task) defines the grace period that is used during reclaim. The consumer that needs the slot has no control over how long it takes to reclaim it.

Tip:

One way to mitigate this situation is to design a consumer hierarchy that places consumers with the same grace period under the same parent consumer. Since both share the quota of the parent consumer, it reduces the possibility of borrowing and lending among consumers with different grace periods.

Disable reclaim when owned slots are proportionally reduced

When the number of slots the consumer owns has been proportionally reduced, you can stop the system from reclaiming some slots from this consumer.

For example, if you configure Consumer X to own 10 slots in a resource group with 100 slots (so X owns 10/100 slots), and the resource group has only 90 slots available at this time, the quota is calculated assuming that X owns only 9 slots.

Then if X is currently using 12 slots and has a new quota of 9, the system should reclaim 3 slots (based on a quota of 9 slots). However, you can configure the system to reclaim only 2 slots (based on the original configuration of 10 owned slots). In this case, the task running on the third slot will be allowed to finish, no matter how long that takes. Then the slot will become available for distribution to another consumer (based on a quota of 9 slots).

To configure this feature, set EGO_DISABLE_RECLAIM_HYBRID_OWN=Y in `ego.conf`.

Feature interactions

This section summarizes the behavior of the feature when interacting with other Symphony features.

Exclusive hosts

The Exclusive host allocation feature can co-exist with the hybrid policy, but the policy will first apply the rules of the Exclusive feature before the rules of the hybrid policy.

Mixing ownership and hybrid policies in one resource group

It is possible to configure both ownership and hybrid policies in one resource group, but only one policy can be configured for a consumer and all its children starting from the first level under the root consumer.

In this type of configuration, the number of slots available for the hybrid policy is the balance from the ownership policy:

- You should configure the owned slots in the hybrid policy so the balance at the root level is equal to 0.
- If the total number of slots in the resource group is *less* than the sum of all owned slots from both policies:
  - the ownership policy consumers own the exact number configured in the ownership policy
  - the hybrid policy consumers all have their owned slots reduced proportionally (the configuration of the hybrid policy does not guarantee that the slots can actually be allocated)
- If the total number of slots in the resource group is *more* than the sum of all owned slots from both policies
  - the ownership policy consumers own the exact number configured in the ownership policy
  - the hybrid policy consumers all have their owned slots increased proportionally (but they are constrained by their limit, which does not change)

Switching policies using time windows

Since Symphony allows resource plan changes to take effect when one time window switches to another, a large number of resources could be reclaimed at once. When the policy switches between ownership and hybrid models, resources in use may be reclaimed, even if they are then redistributed to the same consumers.

For the following example, assume a consumer at time window T1 is configured to use the ownership policy and at time window T2, it is configured to use the hybrid policy.

- If it has lent out slots at T1, the policy will immediately reclaim back all the lent out slots from its borrowers
- If it has borrowed slots at T1, the borrowed slots will be reclaimed.
- If the number of used slots in T1 is more than owned slots in T2, the delta will be reclaimed.

The same thing happens when a consumer tree at time window T1 is configured to use the hybrid policy and time window T2 is configured to use the ownership policy. If the number of used slots in T1 is more than owned slots in T2, the delta will be reclaimed. The borrower's grace period is observed in both cases.

Note:

Note that in both cases, slots borrowed with the ownership policy are independent of slots borrowed with the hybrid policy. The borrower does not need to wait for its borrowed slots from one policy to be released first before it can borrow from another policy. If the borrower has a long grace period, it might temporarily use more slots than you configured in the current active policy, and cause other consumers to be under-allocated. The total number of slots used can exceed the limit during the grace period.

Configure the hybrid scheduling policy

Configure the hybrid policy via the PMC as follows:

1. Select Consumers > Consumers & Plans > Resource Plan
2. In the Time Intervals and Settings dropdown list, select Show Advanced Settings.
3. Expand the Slot allocation policy section. Select Hybrid policy.
4. Enter the number of owned slots, reserve slots, and slots limit for the consumer you want to configure with the hybrid policy.

5. Click Apply.

Prevent unwanted reclaim of owned slots when using the hybrid policy

Configure EGO_DISABLE_RECLAIM_HYBRID_OWN=Y in the `ego.conf` file, when you want EGO to only reclaim the number of slots exceeding the consumer’s owned slots, even though its quota may be less than the number of owned slots. If EGO_DISABLE_RECLAIM_HYBRID_OWN is set to "N", EGO will reclaim the number of slots that exceed the consumer's quota.

Create or modify a resource plan

Make sure you are at the cluster or top level of your tree.

Note:

Before changing the default resource plan, you should export it. Doing this allows you to reset to the default plan, if required, by importing it again.

So that EGO allocates resources how and when you want them, create a resource plan to distribute resources from each resource group to each consumer. The more flexible your plan is, the more flexible your cluster is. For example, allowing lending and borrowing without setting limits helps your cluster react and change to the dynamic needs of your workload. However, you should not enable lending and borrowing for your InternalResourceGroup and ManagementHosts resource groups, nor change the default ownership; the services require a number of owned slots to run the cluster.

The Platform Management Console stores only your active resource plan. You can, however, import a resource plan in XML format to make updates or simply make changes to the resource plan from the Platform Management Console.

A simple default resource plan is already in place to adapt for your own purposes.

Any changes you make and apply to the resource plan in the Platform Management Console are implemented immediately. Do not apply the changes if you do not wish the changes to be immediate. Export the plan instead, and work on the copy until you are ready to import the updates and apply the changes.

1. Locate your resource plan by navigating to Consumers > Consumers & Plans > Resource Plan.

   Your resource plan displays.

   If you have never updated or imported a resource plan, your resource plan shows default settings.
2. Create and manage any time intervals.

   Select Time Intervals and Settings > Insert a time interval or Remove Time Interval: […], where […] is the name of a specific time interval.
3. If you have created resource groups, choose the resource group you want to set the resources for first from the drop down menu.

   You can insert different time intervals for different resource groups.

   The InternalResourceGroup and ManagementHosts resource groups are reserved for system services running your cluster. We do not recommend making changes to these two resource groups without an explicit Platform Support recommendation.
4. Click Expand All.

   Your entire consumer tree expands.
5. Select Time Intervals and Settings > Show Advanced Settings.

   Your advanced settings display, including lending and borrowing.
6. Set your owned slots for each consumer until the balance for each consumer branch in each time period is appropriate for your needs.

   Under most circumstances, the balance should be zero. Under some circumstances (like host scavenging, if available), you may need to allocate more slots than you own.

Keep in mind that the resource plan is hierarchical in nature; you cannot allocate to a leaf consumer more resources than its branch owns.

7. (Optional, but recommended.) For each leaf consumer that has something registered to it, select the options to Lend and Borrow.

   Lending allows a consumer's unused slots to be used by other consumers. Borrowing lets a consumer use unowned or lent slots when they are available.

   Note:

   This step does not apply to the InternalResourceGroup or ManagementHosts resource groups that are reserved for running services that control your cluster. Do not make changes to the consumers that run system services. Do not enable lending or borrowing.

8. (Optional if Lend is checked) Next to Lend, click Details. In the Lend Details dialog box, specify the consumers you want to enable lending for and the maximum number of slots you would like to lend Limit field.

   This number is the maximum number of slots that can be lent out. The remainder are never lent, even if they are not being used. If you leave the field blank, all slots can be lent out.

   You can specify how many slots you want to lend to each individual consumer by clicking Details.

9. (Optional if Borrow is checked) Next to Borrow, specify the maximum number of slots you would like to borrow at any time in Limit.

   This number is the maximum number of slots that are ever borrowed by this consumer at any one time. If you leave the field blank, borrowing is limited only by the amount of resources available in the cluster.

   You can specify how many slots you want to borrow from each individual consumer by clicking Details. Select the consumer you want to enabling borrow for and use the Order column to specify the order in which you want to borrow from specific consumers, where 1 is first. Do not repeat any numbers.

   Note:

   If you modify the borrow order, you must run `egosh ego restart` from the command line.

10. (Optional) Specify the share ratio that applies across consumers at the same level in one branch.

    - If you want sibling consumers to share the resources equally, type 1 for all.
    - If you want one leaf consumer to have twice as many resources as its sibling, type 2 for the first consumer and 1 for the second consumer.
    - If you want one consumer to give up all its borrowed resources when a sibling has demand, specify 0 for the low-priority consumer.

    Your share ratio can be as simple or as complex as you like. Zero and all positive whole numbers are valid entries.

    Note:

    This step does not apply to the InternalResourceGroup or ManagementHosts resource groups that are reserved for running services that control your cluster. Do not make changes to the consumers that run system services; the share ratios should be uniform and there should be no limits.

11. Repeat the resource distribution for all time periods created.

    By default, there is only one time period (00:00 to 24:00).
12. Click Apply to save and make the current settings active.

    If you do not want to make the current changes active, export the resource plan instead.
13. Repeat the entire process for each resource group.

    Switch between resource groups using the drop down list above the plan. Make sure to apply changes before switching to a different resource group (click Apply).

    Note:

    The InternalResourceGroup and ManagementHosts resource groups are special. They are reserved for running system services that control the cluster. If you add your own services, you may need to allocate one or more slots from the ManagementHosts resource group to run workload. We do not recommend changing ownership or enabling lending or borrowing for consumers and services that are created for you by default.

You have modified the default resource plan and created a new plan that efficiently distributes resources across your cluster. Modifications have been applied and the plan is active. You can export this plan and import it again as needed if you want it to apply to specific days of the week only. You can create as many resource plans as you want and import them as needed for a quick change in your distribution.

Feature: Resource plan modification by CLI feature

Use resource plan modification when you want to modify a resource plan using a script.

About resource plan modification by CLI

The EGO CLI Resource Plan Modification feature allows you to modify your resource plan using the command line interface (CLI). Use this feature when you want to modify a resource plan using a script. Benefits of using this feature include triggering the update of resource plan by means of a script. Without this feature, you must log on to the Platform Management Console (PMC) to modify the resource plan.

Scope

Applicability	**Details**
Operating system	Windows
EGO version	EGO 1.2.3 and up
Commands	Applies only to the egosh command
Limitations	The feature is only available upon request
Known issues	None

Behavior of resource plan modification by CLI

After exporting, editing, and saving the resource plan as `C:\ResourcePlan.xml`, `egosh applyresplan C:\ResourcePlan.xml` updates the resource plan:

- Checks user rights.

  Only the Cluster Administrator or the Consumer Administrator of the top-level consumer can update the resource plan using egosh.
- Validates the applied resource plan against the schema.

  The applied resource plan must be a valid resource plan XML file.
- Checks for conflicts between the applied resource plan and the backend resource plan.

  The consumer in the applied resource plan must exist in the backend resource plan.
- Delivers the applied resource plan to the backend and makes it take effect.

  Calls the EGO API to deliver the applied resource plan to the backend.

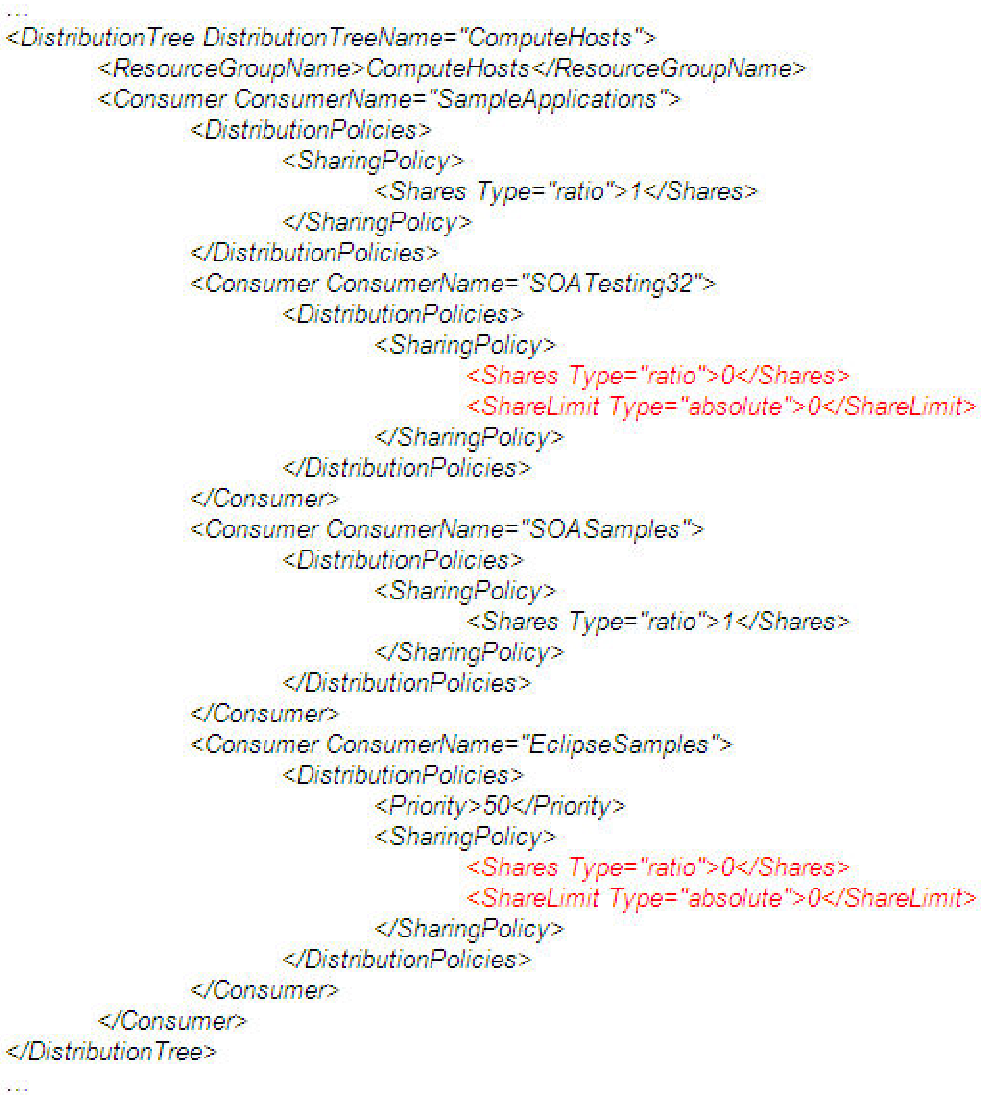
Usage

Syntax	**Description**
consumer applyresplan [-c] [-e *error_log_directory*] *file_path*	Applies the specified resource plan
-c	Only checks the resource plan without applying it.

Syntax	Description
-e	Specifies the directory of the error log. If an error occurs while updating the resource plan, the error messages are appended into the error log in the specified directory. If the directory of the error log is not specified, the system outputs error messages to `stderr` .
file_path	Specifies the path of the resource plan that you want to apply. The resource plan must be a valid resource plan XML file.

Example of usage

1. Prepare the resource plans.
    1. Log on to the Platform Management Console as a cluster administrator.
    2. Export the resource plan and save it as `C:\ResourcePlan.xml`.
    3. Edit ResourcePlan.xml in order to allocate all the resources available to SOASamples, and save it as `C:\ResourcePlan_new.xml`.



```
...
<DistributionTree DistributionTreeName="ComputeHosts">
        <ResourceGroupName>ComputeHosts</ResourceGroupName>
        <Consumer ConsumerName="SampleApplications">
                <DistributionPolicies>
                        <SharingPolicy>
                                <Shares Type="ratio">1</Shares>
                        </SharingPolicy>
                </DistributionPolicies>
                <Consumer ConsumerName="SOATesting32">
                        <DistributionPolicies>
                                <SharingPolicy>
                                        <Shares Type="ratio">0</Shares>
                                        <ShareLimit Type="absolute">0</ShareLimit>
                                </SharingPolicy>
                        </DistributionPolicies>
                </Consumer>
                <Consumer ConsumerName="SOASamples">
                        <DistributionPolicies>
                                <SharingPolicy>
                                        <Shares Type="ratio">1</Shares>
                                </SharingPolicy>
                        </DistributionPolicies>
                </Consumer>
                <Consumer ConsumerName="EclipseSamples">
                        <DistributionPolicies>
                                <Priority>50</Priority>
                                <SharingPolicy>
                                        <Shares Type="ratio">0</Shares>
                                        <ShareLimit Type="absolute">0</ShareLimit>
                                </SharingPolicy>
                        </DistributionPolicies>
                </Consumer>
        </Consumer>
</DistributionTree>
...
```

2. Switch resource plans using egosh in your script.
    1. Log on as a cluster administrator in egosh.

       The credentials are saved in a temp directory. Over the next 8 hours, egosh uses this credential to execute commands and does not need the user's password.
    2. In your script, call `egosh applyconsumer -e C:\ C:\ResourcePlan.xml` and `egosh applyconsumer -e C:\ C:\ResourcePlan_new.xml` to switch resource plans when some event occurs.

Behavior of configuration to modify resource plan modification by CLI

None.

Resource plan modification by CLI commands

Commands to monitor

User	Command	Behavior
Cluster Administrator or Consumer Administrator	From the Platform Management Console: **Consumers & Plans** > **Resource Plan**	Check the resource plan.

Commands to control

User	Command	Behavior
Cluster Administrator or Consumer Administrator	From the Platform Management Console: **Consumers & Plans** > **Resource Plan**	Export the resource plan.
Cluster Administrator or Consumer Administrator	`egosh consumer applyresplan`	Modify the resource plan.

Commands to display configuration

Not applicable.

Commands for submission

Syntax	Description
consumer applyresplan [-c] [-e *error_log_directory*] *file_path*	Applies the specified resource plan
-c	Only checks the resource plan without applying it.
-e	Specifies the directory of the error log. If an error occurs while updating the resource plan, the error messages are appended into the error log in the specified directory. If the directory of the error log is not specified, the system outputs error messages to stderr.
file_path	Specifies the path of the resource plan that you want to apply. The resource plan must be a valid resource plan XML file.

Troubleshooting

Log files

When egosh finds errors while applying the resource plan, it shows detailed messages in stderr. When you specify an error log path, egosh appends error messages into the error log. This is useful when your resource plan is not a valid XML file, in which case the error message may include the whole XML string, which is a large amount of information to display as stderr.

The error log is named egosh.log.%hostname%.

CLI output

- If the resource plan is updated successfully, egosh shows the following messages:

  "The resource plan of consumer %s has been updated for DistributionTree %s"
- If the resource plan fails to update, egosh shows error messages for individual error cases. For example,

  "Your user account has insufficient rights to run the egosh command. Log on as Cluster Administrator or Consumer Administrator"
- If you run "egosh consumer applyresplan" with the "-c" option to check the applied resource plan and the resource plan is valid, egosh shows the following message:

  "The resource plan %s is valid; it is ok to apply the resource plan."

Slot allocation policy

Slot allocation policies offer you greater control over which host slots are allocated to run workload and system services. Symphony supports three slot allocation policies:

- stacked (default)

  Free slots from one host are all allocated first before allocating from another host.
- balanced

  Slots are allocated from the host with the highest number of free slots.
- exclusive

  All slots of each host are allocated to one consumer at a time.

Feature: Balanced Slot Allocation Feature: Allocating CPU Slots Equally Across Hosts

The Balanced slot allocation feature ensures that CPU slots are evenly allocated from all available hosts, and one host is not overloaded while other hosts are sitting idle.

About Balanced slot allocation

Scope

Applicability	Details
Applies to	• Applies to all resource groups.
Operating system	• All OS types supported by the system.
Permissions	• You must be a cluster administrator to change the slot allocation policy.

Default behavior: Stacked slot allocation

By default, the slot allocation policy is Stacked. With this policy, for each allocation, hosts are selected in the order in which they are listed in the resource group during vemkd initialization.

EGO allocates CPU slots from one host until all the CPU slots on the host are used. When all the CPU slots on that host are used, another host is selected and CPU slots from that host are allocated until the all the CPU slots on that host are used, and so on.

Example with ManagementHosts resource group

The following example illustrates the Stacked slot allocation policy for the `ManagementHosts` resource group and the distribution of system services on management hosts.

Default configuration for EGO system services and management hosts:

- Management hosts are configured with 12 CPU slots per CPU
- Out-of-box, there are 8 system services. Each system service takes up 1 slot:

Out-of-box services	Slots to use
ServiceDirector	1

Out-of-box services	Slots to use
WEBGUI	1
WebServiceGateway	1
RS	1
purger	1
`plc`	1
`derbydb`	1
SD	1

If you have Symphony installed, you would additionally have one session manager per application.

Additional services	Slots to use
SSM (one per application)	1

Example details

For this example, you have 3 management hosts in the `ManagementHosts` resource group, 12 CPU slots configured per host.

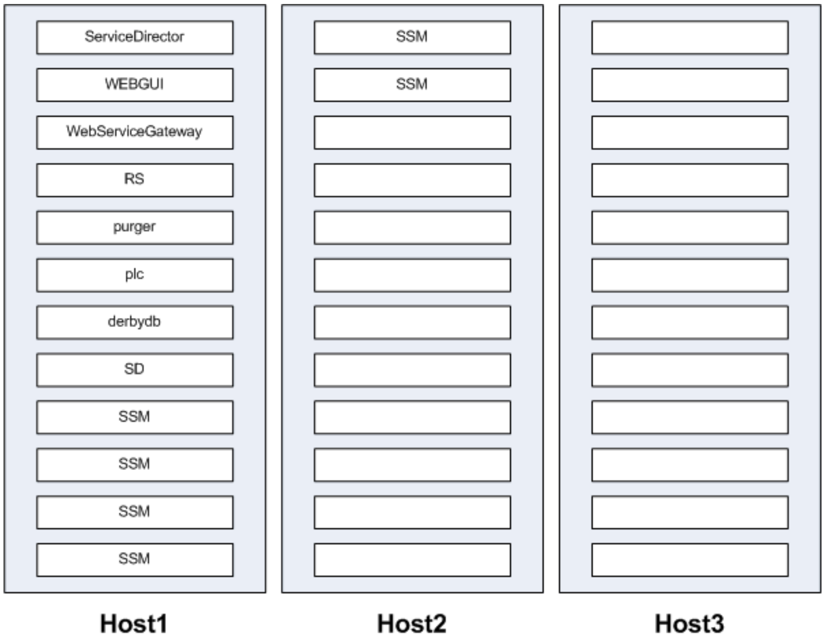
You additionally have 6 Symphony applications, which means 6 session managers. In total, you need 8 CPU slots for system services and 6 more CPU slots for session managers, a total of 14 CPU slots.

With the Stacked slot allocation policy, CPU slots are allocated as follows:

1. To allocate CPU slots for the SD service, EGO evaluates host1, host2, and host3 as listed in the resource group. EGO selects host1 and places SD on host1.
2. To allocate CPU slots for the WEBGUI service, EGO uses available CPU slots on host1.
3. In this way, EGO continues to allocate slots from host1 until all CPU slots from host1 are allocated.

   Once all CPU slots from host1 are allocated, host2 is selected and CPU slots from that host are allocated until all CPU slots from host2 are allocated or until CPU slots have been allocated for all services.

After CPU slots have been allocated for all services, service distribution is as follows on management hosts:



Behavior with Balanced slot allocation

With the Balanced slot allocation policy, for each allocation, hosts are selected from the resource group according to the number of free CPU slots.

Slots are allocated first from the host with the highest number of free CPU slots. When all the CPU slots on that host are allocated, CPU slots are allocated from the next host with the highest number of free CPU slots, and so on.

Since the number of free CPU slots on a host decreases with each allocation, the same host will not be reselected unless it still has the highest number of free CPU slots. As a result, slots are allocated equally from all hosts.

Example with ManagementHosts resource group

The following example illustrates the Balanced slot allocation policy for the `ManagementHosts` resource group and the distribution of system services on management hosts.

Benefits

The benefits of using the Balanced slot allocation policy for the `ManagementHosts` resource group are:

- Performance. Allocating CPU slots from different hosts (and thus starting system services on different hosts) can improve performance during service startup and normal operations.
- Recovery. Should the host fail, recovery time may be reduced because there are fewer system services running on each host.

Default configuration for EGO system services and management hosts:

- Management hosts are configured with 12 CPU slots per CPU
- Out-of-box, there are 8 system services. Each system service takes up 1 slot:

Out-of-box services	Slots to use
ServiceDirector	1
WEBGUI	1
WebServiceGateway	1
RS	1
purger	1
`plc`	1
`derbydb`	1
SD	1

If you have Symphony installed, you would additionally have one session manager per application.

Additional services	Slots to use
SSM (one per application)	1

Example details

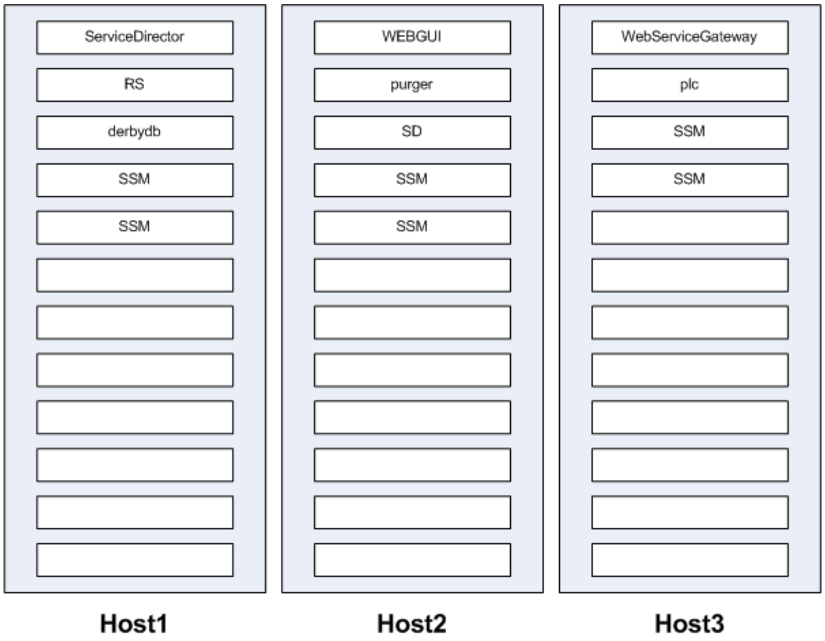
For this example, you have 3 management hosts in the `ManagementHosts` resource group, 12 CPU slots configured per host.

You additionally have 6 applications, which means 6 session managers. In total, you need 8 CPU slots for system services and 6 more CPU slots for session managers, a total of 14 CPU slots.

With the Balanced resource allocation policy, slots are allocated as follows:

1. To allocate CPU slots for the SD service, EGO evaluates host1, host2, and host3 as listed in the resource group. All hosts have the same number of free CPU slots so EGO selects host1 and allocates slots from host1 for SD.
2. To allocate CPU slots for WEBGUI, EGO evaluates the free number of CPU slots on all hosts. Both host2 and host3 have the highest number of free CPU slots, 12, so EGO uses the order in which hosts are listed in the resource group to narrow down its selection. EGO selects host2 and allocates slots from host2 for WEBGUI.
3. To allocate CPU slots for WebServiceGateway, EGO evaluates the free number of CPU slots on all hosts. Host3 has the highest number of free CPU slots (12), so slots from host3 are allocated for WebServiceGateway.
4. In this way, EGO continues to allocate slots from hosts until CPU slots have been allocated for all services.

After CPU slots have been allocated for all services, service distribution is as follows on management hosts:



Balanced slot allocation and failover

Using the `ManagementHosts` resource group example, if a host goes down, slots are allocated for services that were running on the host that went down according to the number of free CPU slots on the remaining hosts. As a result, services are restarted on the remaining hosts according to the number of free CPU slots on the remaining hosts.

When the host that went down comes back up and is available, distribution remains unchanged. Services are not migrated back to the original hosts. As a result, the host that came back up will have the highest number of free CPU slots and will be selected when new services need to be started or restarted on management hosts.

Balanced slot allocation and hosts that belong to multiple resource groups

When the same host belongs to different resource groups, if the different resource groups have different slot allocation policies, the CPU slots belonging to each resource group are allocated according to the policy applying to the resource group.

Configure equal system service distribution among management hosts

Set the Balanced slot allocation policy

By default, system services are started on one management host until all the CPU slots on the host are used, then another host is selected and services are started on that one until all the CPU slots on that host are used, and so on. This can cause some management hosts to be overloaded while other hosts remain idle.

You can, however, configure the system so that system services are evenly distributed on management hosts. You do this by configuring the Balanced slot allocation policy on the `ManagementHosts` resource group.

When the Balanced slot allocation policy is enabled, hosts are selected according to the number of free CPU slots on the host. Since the number of free CPU slots on a host decreases with each allocation, the same host will not be reselected unless it has the highest number of free CPU slots.

1. Select Consumers > Consumers & Plans > Resource Plan.

   The Resource Plan window is displayed.
2. Select the ManagementHosts resource group.

   Resource Plan settings for this host group are displayed.
3. Select Time Intervals and Settings > Show Advanced Settings from the drop-down menu.

   The slot allocation policy is displayed.
4. Click Slot allocation policy to expand and select Balanced. Use free slots from the host with the highest number of free slots.

   You are given an information message because the Balanced slot allocation policy will not affect any system services that have already been started. Only system services that are started from this moment on will be allocated using the Balanced slot allocation policy.

Apply the Balanced slot allocation policy to current allocations

When you change the slot allocation policy, the new policy is only applicable to allocations requests made after the policy was changed.

In some cases, you may want to apply the policy immediately so that system services are redistributed equally among management hosts. You can do this by stopping and restarting the system services. Note that restarting services will affect cluster operations. Refer to the table below for details.

The following is a list of out-of-box services and the effect on your cluster during service restart:

Out-of-box services	Effect of restarting it
ServiceDirector	Services cannot be registered with Service Director.
WEBGUI	The Platform Management Console is temporarily unavailable. Users that are currently connected to the Platform Management Console will need to log on again.
WebServiceGateway	New clients cannot connect to the cluster through the Web Services API.
RS	Package deployment is not available.
purger	No data is purged from the database.
`plc`	Data is pending to be collected.
`derbydb`	Data cannot be stored in the Derby database.
SD	New client applications cannot connect to Symphony. SOAM commands cannot be run.

1. View how services are allocated among management hosts and make a note of which services you want to redistribute among management hosts.

   ```
   egosh service list -l
   ```
2. Restart system services so that slots are allocated from management hosts according to the new allocation policy.

   To restart all system services in the cluster:

   ```
   egosh service stop all
   ```

   ```
   egosh service start all
   ```

To restart each service individually, replace *service_name* with the name of the service:

```
egosh service stop service_name
```

```
egosh service start service_name
```

For example, to restart Service Director:

```
egosh service stop ServiceDirector
```

```
egosh service start ServiceDirector
```

3. View redistribution of services among management hosts.

```
egosh service list
```

4. For each application in your cluster, disable and enable it to restart the session manager.

Important:

When you disable an application, workload is terminated.

```
soamcontrol app disable myapp -s
```

```
soamcontrol app enable myapp
```

5. View redistribution of session managers among management hosts.

```
soamview app
```

Exclusive slot allocation policy

In a cluster made up of multi-core hosts, it is possible that, over time, a host's usage can become fragmented. This can happen when a single-thread task is allocated a multi-core host. In this case, the task occupies one core but the remaining cores are not used. This problem can extend to many hosts in the cluster to the point where there are a lot of free slots but there are no hosts that are totally free.

Here is an example:

Suppose we have two 8-core hosts and two applications: one with 1-core sessions and one with 4-core sessions. If there are five 1-core tasks from App1 occupying 5 of 8 slots on HostA and then a 4-core task is submitted by App2, then it may get the remaining 3 slots on HostA and another 1 slot on HostB. However, this configuration of slots is not usable by the 4-core task in App2.

Exclusive host allocation, which is configured through the Platform Management Console for each resource group in the resource plan, can be used to resolve the resource fragmentation problem.

When Exclusive is set, all the slots of each host in the resource group are assigned to only one consumer at a time. So if there is one slot on a host allocated to one consumer, remaining slots on the same host will not be allocated to other consumers; these slots will only be allocated to this consumer when it has demand. When a consumer reclaims resources from another consumer, EGO enforces the reclaiming of all slots on a host. Note that this behavior may cause a consumer to be allocated less slots than it deserves since a consumer can only be allocated slots on a host when it deserves the whole host. The same principle applies to a consumer that wants to reclaim slots.

When Exclusive is set, if the number of slots on a host is increased, EGO does not allocate the extra slots to the application until existing allocations on the host are released.

Feature interactions

The exclusive slot policy is not compatible with the rusage feature that limits the number of service instances an application can run on a host; refer to *Limit the number of service instances that can run on a host* on page 433.

Configuration to change the slot allocation policy

When you change the slot allocation policy, the change only applies to future allocations, not existing ones. For example, if changing the slot allocation policy for the `ManagementHosts` resource group, any started services remain on hosts from which slots have been allocated and are not redistributed until the services are restarted.

Configuration	Configuration source	Setting	Behavior
Slot allocation policy	From the Platform Management Console Dashboard: 1. **Consumers** > **Consumers & Plans** > **Resource Plan** 2. Select the resource group. 3. Select **Time Intervals and Settings** > **Show Advanced Settings** from the drop-down menu. 4. Click **Slot allocation policy** to expand it. 5. Select the specific policy.	• Stacked. Allocate slots from a single host before allocating from another host. • Balanced. Allocate slots from the host with the highest number of free slots. • Exclusive. Allocate all free slots from a host to one consumer.	• Stacked. Hosts are selected in the order in which they appear in the resource group during vemkd initialization. CPU slots are allocated from a single host until all CPU slots on that host are allocated, before slots are allocated from another host. • Balanced. Hosts are selected according to the number of free CPU slots. The host in the resource group with the highest number of CPU slots is selected first. CPU slots are equally allocated from hosts. • Exclusive. Each host in the resource group is assigned to only one consumer at a time.

Interfaces for slot allocation

Actions to monitor

User	Command	Description
• Any user	egosh resource list -o freeslot	Displays the total number of unallocated CPU slots on each host.
• Cluster administrator	egosh alloc list	Displays the number of allocated slots by resource and resource group.
• Cluster administrator	egosh service list	Displays system services, allocated slots, host name, and resource group so that you can see the distribution of services across hosts.

Actions to control

Not applicable.

Actions to display configuration

User	Command	Behavior
- Cluster administrator	From the Platform Management Console Dashboard: 1. **Consumers** > **Consumers & Plans** > **Resource Plan** 2. Select the resource group. 3. Select **Time Intervals and Settings** > **Show Advanced Settings** from the drop-down menu. 4. Click **Slot allocation policy** to expand it.	- Displays the selected slot allocation policy.

Scenario: Changing resource distribution models

Goal

You change your current resource distribution model (from either full ownership with lending and borrowing or full share with no ownership) to a hybrid model that supports both resource ownership and sharing.

Scenario A: From Ownership to Hybrid model

As a cluster administrator, you have originally set up a resource distribution model that supports full resource ownership. You may have enabled lending and borrowing of resources within the cluster, or you may have set up a "silo" model where there is no borrowing or lending of resources between consumers.

You now wish to change models so that consumers can have access to both owned resources (either their own or those allocated to other consumers) and shared cluster resources (unowned resources that make up the cluster's share pool). You want a more flexible and less prescribed model of resource distribution that responds better to fluctuating workload conditions and number of cluster resources.

Scenario B: From Share to Hybrid model

As a cluster administrator, you have set up a distributed resource sharing model where there are no owned slots, but where each consumer has an assigned share ratio of cluster resources ("share pool").

Your company has a new line of business that requires specific resources (for example, from a certain resource group). The previous full share model, where resources were flexibly allocated depending on workload conditions, consumer priority, and resource availability, cannot guarantee the allocation of the hosts required to satisfy the needs of the new line of business. You want to implement a hybrid model where some resources remain in the share pool, while others are allocated specifically to the new line of business in order to guarantee a minimum level of resource allocation.

At a glance

1. Gather the facts
2. Plan
3. Change the resource plan to support a Hybrid model
4. Add a new consumer to a Hybrid model resource plan
5. View the number of owned and shared slots

Gather the facts

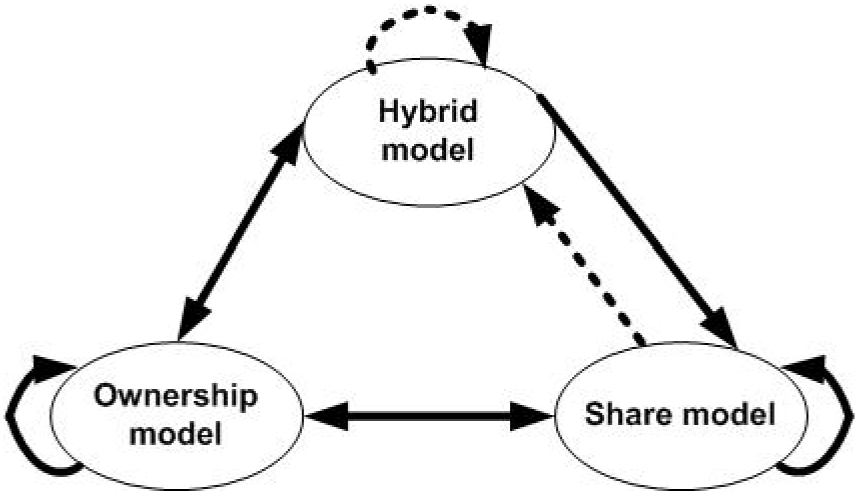
EGO supports three resource distribution models in the resource plan.

Ownership model (“silo”, or with lending/borrowing)	Share model	Hybrid model
This model can support resource distribution ranging from “silo” (where each consumer has a fixed ownership number and there is no lending and borrowing between them), to limited lending/borrowing among consumers, to unlimited lending and borrowing of resources between consumers. **Note:** Share limits are set to zero for all consumers in this model.	In this model, each consumer has a portion (assigned share ratio) of the clusters resources (“share pool”). Each consumer has the potential to use resources from the share pool if other consumers have no competing demand for resources. If there is competition between consumers for resources from the share pool, note the following: **1.** A consumer’s rank is enforced. A consumer with a higher rank than its sibling receives its ratio of share pool resources first. **2.** A consumer’s assigned share ratio is enforced. EGO automatically reclaims the resources from the over-allocated consumer. **Note:** Consumers do not own any slots in this model. Set ownership to 0, as lending and borrowing configurations are of no effect. Ensure share ratios are suitably configured across all consumer branches and for each consumer.	In this model, you can create a resource plan that combines both models (Hybrid model). In this model, EGO maintains a share pool and also supports configured resource ownership by leaf consumers.

The following diagram shows all resource distribution models and potential model migrations. Each arrow represents either a complete migration from one model to another (there are six migration possibilities) or a change within a current resource distribution model.

Note:

The dotted arrows indicate those resource distribution migrations/ changes with notable restrictions. Resource distribution model migrations/changes indicated by solid arrows have no restrictions or associated issues.



There are restrictions for the resource distribution model migrations/changes indicated by dotted arrows.

Restrictions when migrating from a Share to a Hybrid model

When you migrate from a Share model to a Hybrid model, resources that were previously in the share pool are allocated as owned resources to consumers in the resource plan; the number of share pool resources therefore decreases, which may potentially cause difficulties.

In some cases, you may not be able to immediately satisfy the ownership requirements set out in your resource plan when migrating from a Share to a Hybrid model. When the new resource plan comes into effect, and the unallocated share pool resources (in the original Share model) are fewer in number than the configured ownership number in the plan (in the new Hybrid model), then the resource distribution requirements of the plan cannot be fulfilled.

Example: In the Share model, the share pool has 100 slots: 60 slots (resources) are already allocated to consumers running workload units, and 40 slots are available for allocation. Upon implementing a Hybrid model, you intend to reduce the share pool to 50 slots and distribute the remaining 50 slots to three consumers (consumers A, B, and C) with an ownership allocation of 20, 20, and 10 respectively. But after reconfiguring the resource plan, consumer C cannot get its 10-slot allocation. This occurs because there are resources in the share pool that are already allocated. During a resource plan migration from a Share to a Hybrid model, the normal logic for reclaiming share pool resources is not enforced.

Summary: If there are not enough unallocated resources in the share pool at the moment the new Hybrid plan takes effect (during runtime), then consumers are not allocated the planned number of owned resources. Allocated resources from the share pool must first be released back to the share pool by the clustered application managers that are using them (for example, Platform Symphony or LSF) before they can be reallocated as owned resources to a consumer.

Restrictions when making resource plan changes within a Hybrid model

When modifying a resource plan for an existing Hybrid model (where there are both owned resources and unowned share pool resources), there are some issues that may arise.

- If you add a new leaf consumer and allocate resources to it by taking them from the share pool, the consumer’s guaranteed ownership allocation may not be satisfied if the share pool resources are in use by other consumers. If there are not enough unallocated share pool resources to fulfill the ownership requirements in the new resource plan, then the new consumer does not receive its planned resources until sufficient resources are released back into the share pool.

- If you allocate share pool resources as owned resources to a consumer at any level in the consumer tree (top-level, parent, or child), and there are insufficient unallocated share pool resources to fulfill the ownership requirements of the new resource plan, consumers throughout the branch remain unsatisfied. (Top-level consumers are satisfied first with available resources, according to the plan, followed by parents and their children.)

There are no issues if resources taken from the share pool are available (if they are not currently allocated to other consumers).

These use-cases do not have any restrictions or associated issues:

- Adding a new consumer with ownership without changing the number of resources owned by the parent
- Reducing the number of resources owned by a consumer and increasing the share pool or changing the share ratio
- Removing some consumers

Plan

Before beginning, learn how to work with and create a resource plan. For more information, you may want to work through a related scenario Creating a Resource Plan that Responds Dynamically to Consumer Needs, as it contains detailed information and procedures on setting up a hybrid resource plan.

Change the resource plan to support a Hybrid model

You already have a working resource plan that supports either a Share or Ownership model. You may wish to export the existing resource plan and save it for later re-use before beginning your modifications.

When implementing a Hybrid resource plan, you must be sure there are enough unallocated resources in the share pool at the time that the new resource plan becomes effective. Unless specified, each step in the following task relates to migrations from either Share or Ownership models.

1. (Original model type: Share) To reclaim allocated share pool resources, and thereby increase the number of resources available for ownership in the plan, set the Limit in the Model Type: Share section of the Console to **0** for all consumers (Consumers > Consumers & Plans > Resource Plan).

   This forces the allocated resources to immediately return to the share pool. You can then allocate reclaimed resources from share pool to a consumer.

   Note: If you take this approach, a clustered application manager running on EGO (for example, Platform Symphony) may have its workload units disrupted if it is using the share pool resources that get reclaimed. Create a new time interval to avoid this issue:

   a) In the Resource Plan, create at least one new time interval. You want a time interval for the existing Share model resource plan, an interval for the new Hybrid model resource plan, and a short interval (for example, 1 hour) to fall between the Share and Hybrid model plans.

      The short time interval provides a buffer between the two resource plans, and allows currently allocated share pool resources that are running workload units to be released back to the share pool before the Hybrid plan becomes effective. The short time interval must be longer than the grace period for resource reclaim.

   b) In the short time interval that falls between the Share and Hybrid plans, set the Limit in the Model Type: Share section to **0** for all consumers.

   c) In the time interval that follows, set the desired resource policies to support a Hybrid model of distribution.
2. Set or adjust owned slots for each consumer to guarantee a minimum allocation of resources.

   When all available resources are allocated, the cluster balance at the bottom of the Owned Slots section changes to 0.

   Important:

   If you are allocating resources from the share pool to a consumer, ensure there are enough unallocated resources in the share pool at the time that the new resource plan becomes effective to satisfy the consumer's configured ownership.
3. Set or adjust a lend policy for each consumer; enable lending for leaf consumers who own resources.
4. Set or adjust a borrowing policy for each consumer; enable borrowing for leaf consumers who own resources.
5. Set or adjust share ratios for each consumer.
6. Set or adjust the consumer rank for each consumer.
7. Click Apply when all changes the resource plan are complete and you are ready to immediately implement the new plan.

Add a new consumer to a Hybrid model resource plan

You already have a working resource plan that supports a Hybrid model. You may wish to export the existing resource plan and save it for later re-use before beginning your modifications.

1. Create a new consumer.
   a) From Consumers > Consumers & Plans > Consumers, navigate to the location in your consumer tree where you wish to add the new consumer, and then select Create a Consumer from the Global Actions list.
   a) Fill in the consumer properties.
2. Update the resource plan.
   a) Set or adjust owned slots for each consumer to guarantee a minimum allocation of resources.

      When all available resources are allocated, the cluster balance at the bottom of the Owned Slots section changes to 0.

      Important:

      If you are allocating resources from the share pool to the new consumer, ensure there are enough unallocated resources from the share pool at the time that the new resource plan becomes effective. Otherwise, the new consumer does not receive its configured ownership until sufficient resources are released back into the share pool.
   b) Set or adjust a lend policy for each consumer; enable lending for leaf consumers that own resources.
   c) Set or adjust a borrowing policy for each consumer; enable borrowing for leaf consumers that own resources.
   d) Set or adjust share ratios for each consumer.
   e) Set or adjust the consumer rank for each consumer.
3. Click Apply when all changes the resource plan are complete and you are ready to immediately implement the new plan.

View the number of owned and shared slots

For comparison purposes, you can view the number shared slots that a consumer gets from the share pool along with the number of owned slots. Do this through the Console or using the command line.

1. From the Console, click Resources > Monitor Resource Allocation.
2. From the command line, run

```
egosh consumer view consumer_name
```

.

CHAPTER

9

Configuring How Resources Are Shared

Rank consumers in order of priority

You must have created at least one consumer before you can set any priorities (rank). You must either be a cluster administrator or a consumer administrator for this branch to modify a consumer rank.

If you have critical workload to run, you can ensure resources are available by assigning a high rank to a consumer. Note that if all consumers have a high ranking, any advantage one may have over another is nullified. Be selective in assigning a high consumer rank to a consumer.

Consumer rank and the resource group or groups assigned to that consumer work in collaboration. Even if you set a consumer's rank high, the resource group must have the resources available. You can also enable borrowing from other consumers to make sure any unowned resources are assigned to your high ranking consumer.

1. Click Consumers > Consumers & Plans, and then Resource Plan.
2. Select Time Intervals and Settings > Show Advanced Settings.
3. Under ConsumerRank, rank as many consumers as you want.

   Specify any non-negative whole number, where 0 is the highest priority. Priority settings are relative to one another within the resource group.

   If you leave the priority blank, that consumer has no priority over any other consumer (it does not form part of any consumer ordering/sequencing).
4. When finished assigning priority settings, click Apply to save your changes.
5. From the drop down list of resource groups, switch resource groups until you have set consumer priorities for all resource groups within the consumer tree.

You may want to enable and specify details for lending and borrowing for this leaf consumer and its siblings, taking into consideration what priority you have set them.

For example, if you have set a low priority for a consumer, you may wish to enable lending with no limits for it, and then enable borrowing from this consumer in the borrowing details of all other consumers. Doing this maximizes the effectiveness of your resource distribution, lending and borrowing policies, and priority settings. In this example, low priority slots are dynamically lent out to higher priority consumers as required.

Creating a resource plan for lending and borrowing

Some questions and options to consider when creating your resource plan, presented from a consumer's perspective.

Questions for leaf consumers	Summary of options for leaf consumers based on current configuration	
	No	Yes
Do you own resources?	You can still borrow resources. Be sure to enable borrowing. Be sure other consumers agree to lend to you.	A leaf consumer is allocated resources directly from its parent.
As an owner of resources, have you enabled lending?	Your resources can only be used by you. Unused resources are wasted and never lent out.	You can reclaim borrowed resources later. You can also set a lending limit so that only some unused resources get lent, and others remain available to you at all times. You can choose to lend to any consumer that wants to borrow or you can specify which consumers you want to lend to.
Do you want to restrict lending to leaf consumers from the same branch only?	When you set your lending preferences, do not restrict them to leaf consumers from the same branch (siblings).	When you set your lending preferences, list only those leaf consumers from the same branch as preferred consumers that you lend to.
Does your branch have surplus, unowned resources?	Your parent has distributed all of its resources between you and the other leaf consumers in your branch.	Surplus resources can be borrowed by you and other leaf consumers when required. Parent consumers are not able to reclaim surplus resources, but they are returned when the demand by borrowers subsides. Parents may choose to create an "imaginary" child to assign surplus resources to allow for more flexibility (to allow for reclaim and other resource planning).
Does your parent consumer have a shortage of resources?	All leaf consumers are allocated resources to match the resource plan. If demand increases, you can borrow more resources from other consumers who have unused resources available for lending.	Parents who do not have enough resources available to meet the demands of their children allocate what they do have to their highest priority child. Once its demands are met, the next highest priority child receives what is left of the parent's resources, and so on.

Questions for leaf consumers	Summary of options for leaf consumers based on current configuration	
	No	Yes
Do you have an unsatisfied demand?	If you have unused resources with lending enabled, they can be borrowed by other consumers experiencing demand.	If you are using all of your owned resources and you still require more, you can borrow surplus resources from other leaf consumers or from the branch-level. Once you have exhausted borrowing and sharing options, you can reclaim those resources that you have lent to other consumers*. If there is a reclaim grace period set by the borrower of your resources, you may have to wait until the grace period expires or until the borrowing consumer finishes running workload units before getting it back. **Tip:** *This default behavior can be changed so that owned resources get reclaimed before a consumer attempts to borrow resources from other lending consumers.
Are there several consumers with a simultaneous demand?	Demand is low, with no competition for resources. Consumers are using their owned resources, or borrowing unused resources without dispute.	If you and a sibling both have an unmet demand for resources, than the share ratios for each of you are considered before your parent allocates surplus resources; if you have a higher share ratio, you receive more of the surplus resources. You may have a maximum limit set for the number of resources you can receive, despite your share ratio. If there is competition between siblings for resources belonging to another consumer branch, the leaf consumer with the highest consumer rank has its demand satisfied first. In cases where the consumer rank is the same, borrowing preference are considered. For example, if the lender is configured to show a higher borrowing preference towards you, then you have your resource demands satisfied first.

Questions for leaf consumers	Summary of options for leaf consumers based on current configuration	
	No	**Yes**
Are you borrowing resources?	Demand may be low. Ensure you enable borrowing in case you have an unsatisfied demand in future.	Borrowing is on a first-come first-served basis. If there are competing borrowers for a consumer's resources, and you are the first one to request them, you get as many as required to meet your demand. Any remaining resources go to the borrowers in line behind. If there are no resources left, competing borrowers must wait for you to release your borrowed resources (once your demand subsides). As a borrower, there is the possibility that any running workload units may be interrupted if the lender of your resources reclaims them. You have the option of setting a reclaim grace period to avoid interrupting any workload units running on a borrowed resource. Resources meeting the demand requirements of the lender are reclaimed according to consumer rank, from lowest to highest.
As a borrower, have you set a grace period to extend how long you can use borrowed resources before they are reclaimed?	If you are a borrower, your borrowed resource gets reclaimed as soon as the lender has demand. Any client work running on the resource is interrupted.	If you are a borrower and the lender wants their resource back to meet their own demand, any running workload units continue to run uninterrupted until the reclaim grace period expires, or until the client releases the resource (whichever comes first).
As a borrower, do you have a borrowing limit?	You can borrow as many owned resources available for sharing as you need to meet your resource demands.	The number of total resources you can use (including owned, borrowed, and shared resources) has a set limit. If your demand is great, you may not be allowed to satisfy it completely.
Are your client's workload units running as expected? Are your resource demands consistently being met?	• Ensure that the entire consumer branch in the consumer tree is adequately resourced. • Check that priority levels are not all set the same (all "low" or all "high"). • Confirm that the share ratio between sibling leaf consumers is appropriate. • Make sure borrowing and lending are enabled.	The resource plan is effective.

Change the default resource allocation policy in conjunction with resource plan

You must be a cluster administrator.

EGO systematically allocates resources according to a default plan. A key component of this plan is that consumers who experience demand borrow resources from other lending consumers before they reclaim any of their lent resources.

You can create a resource plan and change the resource allocation behavior so that owned resources get reclaimed by consumers before they are borrowed or allocated from elsewhere. This is done in coordination with an effective resource plan that properly reflects your business requirements.

1. Create an appropriate resource plan (Consumers > Consumers & Plans > Configure Resource Plan).

   If you have never created, updated, or imported a resource plan, the base plan is your only plan and it has default settings. See the topic Create or modify a resource plan for a detailed procedure. Find a summary below:

   a) Set the date and time as well as the frequency of occurrence.
   b) If you have created resource groups, choose the resource group you want to set the resources from the drop-down menu (for example, Resource Group: ComputeHosts).
   c) Select Show Advanced Settings.
   d) Click Expand All.
   e) Set your owned slots for each consumer until the balance for each consumer branch in each time period is appropriate for your needs.

      Under most circumstances, the balance should be zero. Under some circumstances (like host scavenging, if available), you may need to allocate more slots than you own.
   f) Rank your consumers, remembering that resources are reclaimed based on rank (those leaf consumers with a lower consumer rank are reclaimed before consumers with a higher rank).

      Specify any positive whole number, where 1 is the highest priority. Priority settings are relative to one another within the resource group. If you leave the priority blank, that consumer has no priority over any other consumer (it does not form part of any consumer ordering/sequencing).
   g) For each leaf consumer that has something registered to it, select the options to Lend and Borrow.

      Lending allows a consumer’s unused slots to be used by other consumers. Borrowing lets a consumer use unowned or lent slots when they are available.
   h) (Optional: if Lend is checked) Next to Lend, click Details. In the Lend Details dialog box, specify the maximum number of slots you would like to lend in the Total lend limit field.
   i) (Optional: if Borrow is checked) Next to Borrow, specify the maximum number of slots you would like to borrow at any time in Limit.
   j) (Optional) Specify the share ratio that applies across consumers at the same level in one branch.

      - If you want sibling consumers to share the resources equally, type 1 for all.
      - If you want one leaf consumer to have twice as many resources as its sibling, type 2 for the first consumer and 1 for the second consumer.
      - If you want one consumer to give up all its borrowed resources when a sibling has demand, specify 0 for the lower-priority consumer. Note that in doing this, the consumer does not receive any resource from the share pool.

   k) Click Apply to save and make the current settings active.

      If you do not want to make the current changes active, export the resource plan instead.

2. Set a reclaim grace period and rebalance behavior for a selected consumer:
   a) Click Consumers > Consumers & Plan.
   b) Click a top-level consumer from the consumer tree.
   c) Click Consumer Properties.
   d) Specify a reclaim grace period to apply when a resource gets reclaimed by its owner.
   e) Check the Rebalance when resource plan changes or time interval changes box if you want EGO to “rebalance” or reset to the originally configured resource plan whenever a time interval change occurs (when there is a change in ownership of resources) or when the resource plan changes.
   f) Click Apply.
3. Change the default reclaim behavior:
   a) Click Cluster > Summary > Cluster Properties.
   b) Specify the resource allocation behavior for the cluster by checking the appropriate boxes in the section Specify resource allocation behavior.
      - To allow the share pool to reclaim resources from an over-allocated consumer, ensure Reclaim shared resources is checked (default).
      - To allow a leaf consumer to reclaim its resources before borrowing from another consumer, check Reclaim lent resources before borrowing.
   c) Click Apply.

EGO continues to allocate resources according to your configured resource plan, but now considers systematically allocating resources in a different order.

Configure EGO to enforce share ratio at parent level

Share ratios are enforced at the leaf level by default. To change the configuration so that share ratios are enforced at the parent level, you need to set a parameter in `ego.conf`.

1. Open `ego.conf`.
   - Windows: `$EGO_CONFDIR\ego.conf`
   - Linux/UNIX: `$EGO_CONFDIR/ego.conf`
2. Set the following parameter in the share directory:

   EGO_PARENT_QUOTA=**`Y`**
3. Save and close `ego.conf`.
4. Shutdown and then restart the cluster (vemkd must be restarted after configuring `ego.conf`).

Feature: Resource reclaim

Resource reclaim—a feature of Platform Symphony's borrowing, lending, and sharing functionality—ensures that consumers can take back their deserved shared or lent resources as needed to meet workload demand.

This is not applicable to Symphony DE.

Scope

Applicability	Details
Operating system	• All host types supported by the Symphony system
Exclusions	• Does not apply to Symphony DE, which does not have resource lending, borrowing, and reclamation

About resource reclaim

Purpose of resource reclaim

Resource reclaim provides a way for the system to reallocate borrowed or shared resources to a consumer when the consumer has workload demand under any of the following conditions:

- A lending consumer has workload demand that requires slots owned by the lending consumer
- Share ratios are configured, and an under-allocated consumer (a consumer that is not currently using its *deserved* number of shared slots) has workload demand that requires the use of more slots
- A time based resource plan has time intervals that change the number of owned resources, share ratios and limits, borrowing and lending policies, and borrowing and lending limits for one or more consumers

The system does not always return the same resource that the consumer originally lent. If workload is running on a borrowed resource, the system could reclaim a different physical resource (that meets the resource requirements) from the borrower and allocate that resource to the lending consumer in place of the original resource.

Benefits of resource reclaim

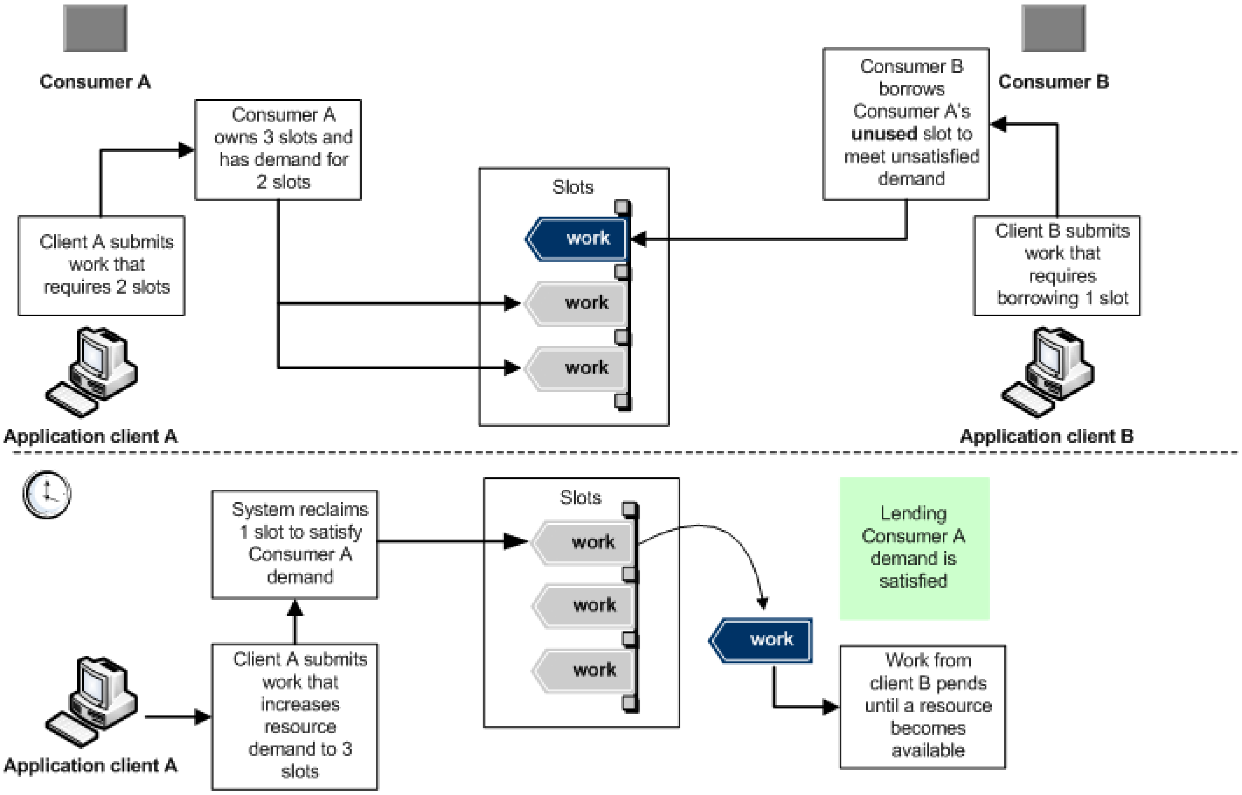
The following illustrations show how the resource reclaim feature works when borrowing, lending, or sharing are enabled.

Important:

Resource reclaim is enabled by default whenever borrowing and lending are enabled. You cannot disable resource reclaim for borrowed or lent resources.

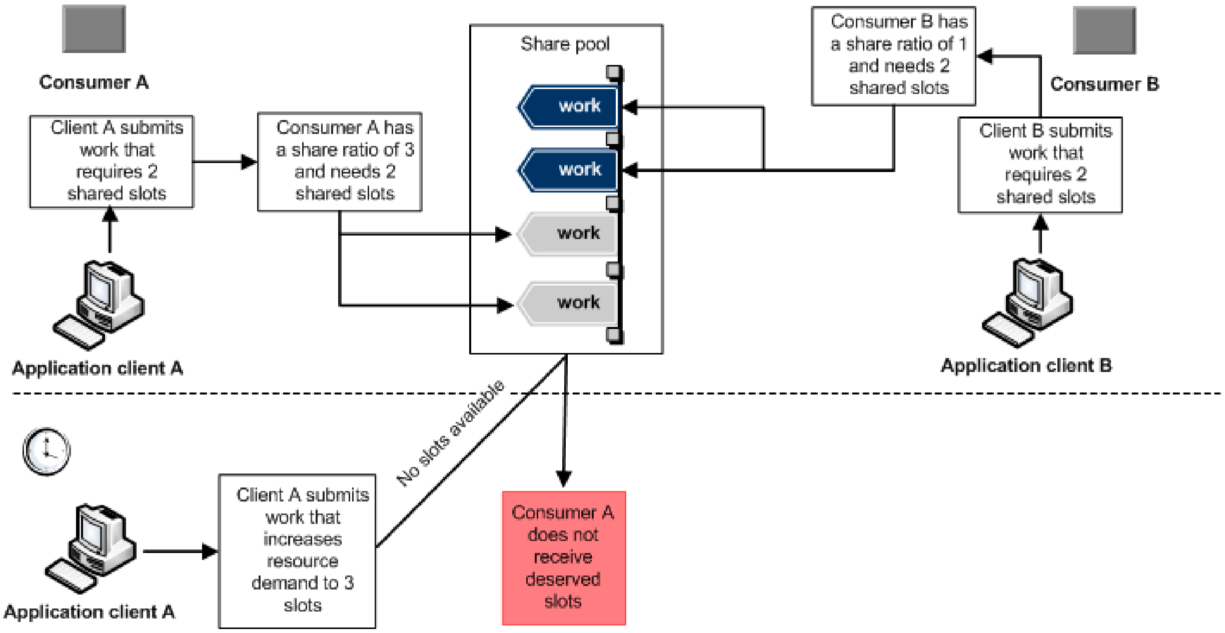
How resource reclaim works for borrowing and lending

You can choose to enable borrowing and lending for owned resources. When you enable borrowing and lending, resource reclaim is always enabled.


Consumer A
Consumer A owns 3 slots and has demand for 2 slots
Client A submits work that requires 2 slots
Application client A
Slots
work
work
work
Consumer B borrows Consumer A's unused slot to meet unsatisfied demand
Consumer B
Client B submits work that requires borrowing 1 slot
Application client B
System reclaims 1 slot to satisfy Consumer A demand
Client A submits work that increases resource demand to 3 slots
Application client A
Slots
work
work
work
work
Lending Consumer A demand is satisfied
Work from client B pends until a resource becomes available

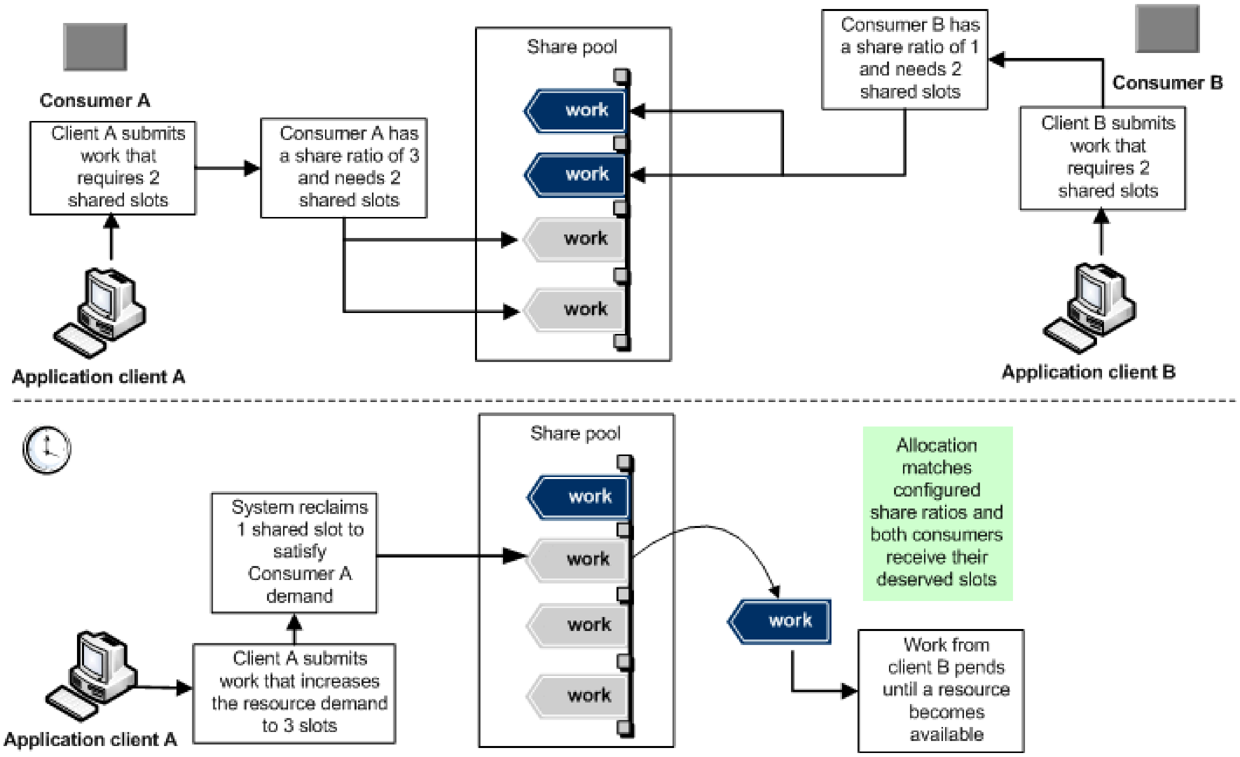
Without resource reclaim for sharing (feature not enabled)

In this example, the share ratio is 3:1. Consumer A deserves 3 times the number of slots as Consumer B.


Consumer A
Client A submits work that requires 2 shared slots
Consumer A has a share ratio of 3 and needs 2 shared slots
Application client A
Share pool
work
work
work
work
Consumer B has a share ratio of 1 and needs 2 shared slots
Consumer B
Client B submits work that requires 2 shared slots
Application client B
No slots available
Client A submits work that increases resource demand to 3 slots
Application client A
Consumer A does not receive deserved slots

With resource reclaim for sharing (feature enabled)

In this example, the share ratio is 3:1. Consumer A deserves 3 times the number of slots as Consumer B.


Consumer A
Client A submits work that requires 2 shared slots
Application client A
Consumer A has a share ratio of 3 and needs 2 shared slots
Share pool
work
work
work
work
Consumer B has a share ratio of 1 and needs 2 shared slots
Consumer B
Client B submits work that requires 2 shared slots
Application client B
Share pool
work
work
work
work
System reclaims 1 shared slot to satisfy Consumer A demand
Application client A
Client A submits work that increases the resource demand to 3 slots
Allocation matches configured share ratios and both consumers receive their deserved slots
work
Work from client B pends until a resource becomes available

Service instance interrupt handling

Resource reclaim is enabled whenever you enable lending or borrowing for leaf consumers that own resources. By default, the system:

- Immediately sends an interrupt event to the service to notify it of the pending reclaim.
- Allows the service the number of seconds specified in the reclaim grace period to complete processing before terminating the service instance. Tasks that were running on the service instance before it was killed are requeued to their respective sessions. The default grace period is 0 seconds.
- After the reclaim grace period expires, EGO allows 120 seconds leeway time for the return of any reclaimed resources. This is to account for network overhead and other considerations.

The onServiceInterrupt service handler method provides the most effective way to manage an interruption caused by resource reclaim. Use of this method ensures that the service instance receives immediate notification of a pending interruption.

During a reclaim, the service interrupt indicates how much time the service instance takes to complete current running service method and the service instance to clean up. 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.

If a task is running and the Invoke method completes during the applied reclaim 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 applied reclaim grace period expires, the service instance on which the task is running is terminated and the task is requeued.

Another but less effective way to manage an interruption is for the service instance to periodically call the getLastInterruptEvent method for interrupt events. With this method, the service instance polls and will not immediately detect the interrupt. While the service instance is polling, the reclaim grace period is expiring, and the service instance will have less time to return a result or shut down gracefully.

Configuration to enable resource reclaim

Borrowing and lending with respect to reclaim

Resource reclaim of borrowed resources is always enabled if you configure borrowing and lending at the consumer level. Borrowing and lending can only be configured at the leaf consumer.

Configuration source	Setting	Behavior
Platform Management Console: **Consumers** > **Consumers & Plans** > **Resource Plan > Show Advanced Settings > Expand All**	**Owned Slots**=*integer*	• Specifies a number of slots owned by a leaf consumer. The leaf consumer is guaranteed to receive this number of slots, provided that the consumer has enough demand. If a consumer's owned slots are lent to a borrowing consumer, and the lending consumer has workload demand, the system initiates a reclaim of the owned slots.
	For the lending consumer: • **Lend** checkbox selected • **Details**: • **Lend** checkbox selected for the consumer to lend to	• Enables the consumer to lend resources to the specified consumer(s) • The specified consumer(s) must have borrowing enabled and specify the lending consumer.
	For the borrowing consumer: • **Borrow** checkbox selected • **Details**: • **Borrow** selected for the consumer to borrow from	• Enables the consumer to borrow resources from the specified consumer(s) • The specified consumer(s) must have lending enabled and specify the borrowing consumer.

Share pool and share ratio

Resource reclaim for shared resources is enabled by default once you configure a share pool and share ratios for at least one consumer branch.

Configuration source	Setting	Behavior
Platform Management Console: **Consumers** > **Consumers & Plans** > **Resource Plan > Show Advanced Settings > Expand All**	**Owned Slots**=*integer*	• Specifies a number of slots owned by a leaf consumer. The leaf consumer is guaranteed to receive this number of slots, provided that the consumer has enough demand. If a consumer's owned slots are lent to a borrowing consumer, and the lending consumer has workload demand, the system initiates a reclaim of the owned slots. • Any unowned slots constitute a "share pool" for allocation to leaf consumers with unsatisfied demand.
	Share Ratio selected and *integer* specified as a value	• Sets the relative share ratios within a share pool. • If you specify 0 for a consumer, that consumer gives up its share of the share pool when a sibling has demand. A consumer with a share ratio of 0 does not receive any resources from the share pool.
Platform Management Console: **Cluster > Summary > Cluster Properties > Specify resource allocation behavior**	**Reclaim shared resources** selected	• When selected (the default setting), the share pool reclaims resources from a consumer that is using more slots than it deserves based on its share ratio to meet the demands of a competing consumer with a higher share ratio.

Resource reclaim behavior

Order of resource reclaim (consumer level)

Consumers reclaim resources in the following order, regardless of a consumer's history of resource usage:

When the system reclaims …	Then reclaim occurs in the order of…	Example
Borrowed resources	Resource requirements, determined by the resource group associated with the consumer.	If the lending consumer needs a Windows slot with a certain amount of available memory, the system looks first for an analogous resource to reclaim.

When the system reclaims …	Then reclaim occurs in the order of…	Example
Shared resources	Relative consumer rank, configured in the Resource Plan. Consumer rank is an optional setting. A rank of 0 is the highest rank and larger numbers indicate a lower rank. The system reclaims resources from the lowest ranking consumer first.	The system first reclaims resources from a consumer with rank 50, and then reclaims resources from a consumer with rank 25.
	• By default, the system enforces share ratios at the level of the leaf (child) consumers. If your system is configured to enforce share ratios at the parent level, the system reclaims resources from the parent consumer.	Consumer A is a child consumer of Parent A. Parent A and Parent B are siblings. With share ratio enforced at the parent level, Parent A shares 10 slots with Parent B. Parent B is running workload on 5 slots obtained from Parent A's share. If Consumer A has unsatisfied demand for 2 slots and all of Parent A's slots are allocated, the system reclaims 2 slots from Parent B to allocate to Parent A.

Order of resource reclaim (resource level)

When the system must reclaim a resource from a consumer, and there are multiple possibilities for which resource could be reclaimed, these steps describe how your configuration choices help to determine exactly which task will be interrupted and which resource will be reclaimed.

Session importance (preemption rank or session priority) and preemption criteria are always potential influences, but the selective reclaim configuration is the most important parameter because it determines whether the other parameters can influence host selection or not. Note that selective reclaim can only be enabled if "Optimized for application specified conditions" (default setting) is configured through the PMC. If selective reclaim is disabled, the system will still select the "best" slot on a host, but it may appear that resource selection happens at random because there is no effort to select the "best" host among multiple candidates.

The system chooses the resource using the following logic.

1. Consider selective reclaim configuration.
    1. If selective reclaim is disabled, reclaim resources as quickly as possible, with minimum overhead. This is the default.

       EXAMPLE: if multiple hosts in the consumer could meet the resource requirement, the system selects any one at random.
    2. If selective reclaim is enabled, reclaim resources from the less important sessions first. This option has greater overhead.

       EXAMPLE: if multiple hosts in the consumer could meet the resource requirement, the system selects all candidate hosts.
2. For proportional or minimum services scheduling, consider preemption rank. For priority scheduling, consider session priority instead of preemption rank.
    1. With proportional or minimum services scheduling:

       From the host or hosts selected, select the least important session, according to preemption rank.

       If multiple sessions have equal low rank, select all candidate sessions.

    If the resource requirement is for an exclusive host, treat all sessions on a host as if they had the same rank as the most important session using the host.

    2. With priority scheduling:

       From the host or hosts selected, select the least important session, according to session priority.

       If multiple sessions have equal low priority, select the most recently started session.

       If the resource requirement is for an exclusive host, treat all sessions on a host as if they had the same priority as the most important session using the host.

3. Consider preemption criteria.

    1. If the criteria is MostRecentTask, reclaim resources from the most recently submitted tasks first.

       EXAMPLE: from one or more sessions, the system selects the most most recently started task and reclaims the resource it is using.

       If multiple tasks have the same run time, the system selects any one at random.

       If multiple tasks run on a slot, consider the cumulative run time of all tasks using the slot.

    2. If the criteria is PolicyDefault, the behavior changes depending on the scheduling policy. This is the default setting for the parameter.

       - With proportional or minimum services scheduling:

         The default is to reclaim resources from the most over-allocated sessions first. This is the option with minimum overhead.

         EXAMPLE: from multiple sessions, the system selects the most over-allocated session, and reclaims a resource it is using (task selection is random).

         If multiple sessions are equally over-allocated, the system selects any one at random.

         If no session is over-allocated, select the least under-allocated instead.

       - With priority scheduling:

         The default is to selects a task from a session with the lowest priority, followed by tasks from the last started session. This is the option with minimum overhead.

Selective reclaim considerations

An application may be a candidate for selective reclaim when it may need to borrow slots from other consumers and has critical or long running tasks that you do not want interrupted.

Important:

Selective reclaim will not take effect if **Reclaim optimization** is configured as **Optimized for standby service** in the PMC.

Here are some considerations when using selective reclaim.

- Are there any critical tasks in the application? If the answer is yes, configure a high preemption rank for critical sessions to protect critical tasks from being interrupted. Otherwise, leave all preemption ranks at the same level. (This only applies to proportional or minimum service policies. For the priority scheduling policy, the session priority is used.)
- If there are long running tasks (not critical ones), set preemption criteria to MostRecentTask so that when reclaim happens, the CPU time of long running tasks is not lost.
- If all the tasks are short running, set preemption criteria to default for better SSM performance.

Consumer demand

Consumers with workload demand can have lent resources reclaimed for them. When the system reclaims a resource, the system interrupts the borrower's tasks running on the reclaimed resource. The reclaim grace period allows time for a task running on a borrowed slot to complete before the resource returns to its owner. To avoid being requeued, tasks must exit within the reclaim grace period.

By default, the system reclaims owned resources only after attempting to satisfy demand by borrowing resources from other lending consumers or from the share pool. You can change this behavior so that the system reclaims owned resources before allocating borrowed or shared resources.

Time interval transitions

With a time based resource plan that specifies different values for ownership, lend and borrow limits, share ratios and limits, or total slots in the share pool, a transition from one time interval to the next can trigger resource reclaim. By default, the system enforces ownership and limits when the new time interval takes effect. The following examples illustrate how time interval changes trigger resource reclaim:

When…	The behavior is…	Example
A consumer's ownership increases for the new time interval, lending and borrowing are not configured, and another consumer is using more than its deserved resources	The system reclaims slots whether or not consumers have unsatisfied demand.	**1.** Consumer A owns 10 slots between 8:00 a.m. and 5:00 p.m. and 25 slots between 5:01 and 11:49 p.m. **2.** At 5:01 p.m., Consumer B is using more than its deserved slots. **3.** At 5:01 p.m., the system reclaims 15 slots to allocate to Consumer A.
A consumer's ownership decreases for the new time interval, and lending and borrowing are not configured	The system reclaims the number of slots required to conform to the ownership values configured for the new time interval, whether or not other consumers have unsatisfied demand.	**1.** Consumer A owns 10 slots between 8:00 a.m. and 5:00 p.m. and 5 slots between 5:01 and 11:49 p.m. **2.** Consumer B owns 5 slots between 8:00 a.m. and 5:00 p.m. and 10 slots between 5:01 and 11:49 p.m. **3.** At 5:01 p.m., the system reclaims 5 slots from Consumer A, even if Consumer A has unsatisfied demand, and allocates 5 slots to Consumer B.

When…	The behavior is…	Example
A consumer's ownership decreases for the new time interval, borrowing and lending for the consumer are configured, and a lending consumer has slots available	The system reclaims the number of slots required to conform to the ownership values configured for the new time interval, and then the consumer borrows available resources; the resource status changes from owned to borrowed.	**1.** Consumer A owns 10 slots between 8:00 a.m. and 5:00 p.m. and 5 slots between 5:01 and 11:49 p.m. **2.** Consumer B owns 5 slots between 8:00 a.m. and 5:00 p.m. and 10 slots between 5:01 and 11:49 p.m. **3.** At 5:00 p.m., Consumer A has workload running on 10 slots and Consumer B has workload running on 5 slots. **4.** At 5:01 p.m., the system reclaims 5 slots from Consumer A, even if Consumer A has unsatisfied demand, and allocates 5 slots to Consumer B. **5.** Consumer A is configured to borrow from Consumer B, and Consumer B is configured to lend to Consumer A. **6.** Consumer B has no demand for the 5 reclaimed slots. Consumer A borrows 5 slots from Consumer B.
A consumer's lend limit decreases for the new time interval	The system reclaims the number of slots required to conform to the new lend limit whether or not the consumer has unsatisfied demand.	**1.** Consumer A has a lend limit of 10 slots between 8:00 a.m. and 5:00 p.m. and 5 slots between 5:01 and 11:49 p.m. **2.** Consumer B borrows 10 slots from Consumer A. **3.** At 5:01 p.m., the system reclaims 5 slots from Consumer B and allocates them to Consumer A.
A consumer's borrow limit decreases for the new time interval	The system reclaims the number of slots required to conform to the new borrow limit, whether or not the lending consumer has unsatisfied demand.	**1.** Consumer A has a borrow limit of 10 slots between 8:00 a.m. and 5:00 p.m. and 5 slots between 5:01 and 11:49 p.m. **2.** Consumer A borrows 10 slots from Consumer B. **3.** At 5:01 p.m., the system reclaims 5 slots from Consumer A to return to Consumer B.
A consumer's share limit decreases	The system reclaims the number of slots required to conform to the new share limit, whether or not a competing consumer has unsatisfied demand.	**1.** Consumer A has a share limit of 10 slots between 8:00 a.m. and 5:00 p.m. and 5 slots between 5:01 and 11:49 p.m. **2.** A share pool is configured for the consumer branch (the parent consumer and its children). **3.** At 5:01 p.m., the system reclaims 5 slots from Consumer A to return to the share pool.

When…	The behavior is…	Example
The total number of slots in the share pool decreases	The system reclaims the number of slots needed to maintain share ratios whether or not a competing consumer has unsatisfied demand.	**1.** Consumers A and B each have a share ratio of 1. **2.** The consumer branch owns 10 slots between 8:00 a.m. and 5:00 p.m. and 4 slots between 5:01 and 11:49 p.m. **3.** At 5:00 p.m., Consumer A runs workload on 5 slots, and Consumer B runs workload on 5 slots. **4.** At 5:01 p.m., consumers A and B each return 3 slots to the share pool. **5.** During the new time interval, Consumer A runs workload on 2 slots and Consumer B runs workload on 2 slots.

Configuration to modify resource reclaim behavior

Configuration to modify the reclaim grace period

You can configure a different reclaim grace period behavior for each consumer.

Important:

The borrowing consumer determines the reclaim grace period. When you configure borrowing and lending, ensure that the lending consumer can wait for the maximum reclaim grace period configured for all of its borrowing consumers.

Configuration source	Setting	Behavior
Platform Management Console: **Consumers > Consumers & Plans > *consumer_name* > Consumer Properties > Reclaim behavior**	**Reclaim grace period**= *integer* **Seconds | Minutes | Hours**	• Specifies the wait time before the system interrupts workload running on a borrowed or shared host to reclaim the resource. • To reclaim resources almost immediately, specify 0 seconds. • If you leave the reclaim grace period blank or specify 0, the system uses a default grace period of 0 seconds. • As a best practice, you should specify a realistic value that allows tasks from all of your applications enough execution time and time to clean up to avoid unnecessary interruption. Consider both the typical length of a workload unit run by a borrowing consumer and the urgency of workload demand from the lending consumer.

Configuration to modify system rebalancing behavior

You can configure system rebalancing behavior for each consumer.

Note:

Child consumers do not inherit the value set for the parent consumer.

Configuration source	Setting	Behavior
Platform Management Console: **Consumers > Consumers & Plans > *consumer_name* > Consumer Properties > Reclaim behavior**	**Rebalance when resource plan changes or time interval changes** selected	- (Default setting) Enforces ownership, share ratios, and borrowing, lending, and share limits for this consumer when the resource plan is changed or the new time interval takes effect, regardless of consumer demand. - If corresponding lending and borrowing consumers have different rebalancing settings (one is selected and the other is deselected), the consumer with an over-allocation determines which setting the system uses, which determines whether rebalancing occurs.
	Rebalance when resource plan changes or time interval changes deselected	- When deselected, the system waits until borrowed resources are returned before enforcing new ownership, share ratios, and borrowing, lending, and share limits for this consumer. Note that due to potential interrelationships of consumers, this setting can impact the enforcement of new ownership, share ratios, and borrowing, lending and share limits for other consumers that must wait for resources to be returned. It is recommended that all consumers in a resource group have the same rebalancing setting.

Configuration to modify reclaim behavior for shared resources

You can configure whether the system reclaims shared resources or waits until consumers release shared resources after completing workload tasks, and whether to enforce share ratios at the parent level.

Configuration source	Setting	Behavior
Platform Management Console: **Cluster > Summary > Cluster Properties > Specify resource allocation behavior**	**Reclaim shared resources** selected	- (Default setting) Enables the system to reclaim resources from an over-allocated consumer when a consumer with a higher share ratio has unsatisfied demand.
	Reclaim shared resources deselected	- When deselected, the system does not reclaim shared resources.
`ego.conf`	EGO_PARENT_QUOTA=**Y**	- Enforces share ratios at the parent level, which allows a leaf (child) consumer to have resources reclaimed from another consumer branch, based on the parent consumers' share ratios. By default `EGO_PARENT_QUOTA` is set to **N**. - You must restart EGO on the master and all master candidates after modifying `ego.conf`.

Configuration to modify reclaim behavior for owned resources

By default, consumers borrow resources before their owned resources are reclaimed. You can modify this behavior so that lent resources are reclaimed before borrowing resources from another consumer. This is useful when a consumer's owned resources have specific characteristics required to run the consumer's workload, or when borrowing from a different consumer branch incurs costs based on charge-back policies at your site.

Configuration source	Setting	Behavior
Platform Management Console: **Cluster > Summary > Cluster Properties > Specify resource allocation behavior**	**Reclaim lent resources before borrowing** selected	- Enables reclaim of owned resources before borrowing resources from other consumers.
	Reclaim lent resources before borrowing deselected	- (Default setting) When deselected, consumers with unsatisfied demand borrow resources from other consumers before having their owned resources reclaimed.

Configuration to enable selective reclaim

By default, the system will reclaim resources as quickly as possible, with minimum overhead. You can modify this behavior so that the system considers the relative importance of running work and reclaims resources from less important sessions first.

Configuration source	Setting	Behavior
Platform Management Console: **Symphony Workload > Configure Applications** Open the application profile and edit the **General Settings** section.	Enable Selective Reclaim = true	- From all the suitable hosts in the consumer, consider session importance (preemption rank or session priority) and preemption criteria to determine which resource to reclaim.
	Enable Selective Reclaim = false	- (Default setting) From all the suitable hosts in the consumer, pick a host at random and reclaim a resource from that host. Session importance (preemption rank or session priority) and preemption criteria determine which slot on the host is chosen.

Configuration to modify preemption rank

This parameter is ignored if priority scheduling is used.

By default, all sessions are considered to be of equal importance when the system is reclaiming resources. You can modify this behavior by ranking sessions in order of importance when you create a new session. When sessions have different ranks, the system may reclaim resources from the low-ranking sessions first.

Preemption rank is similar to session priority but it cannot be changed after the session has started, and it is not used when the system has to allocate resources, only when it has to reclaim them.

Configuration source	Setting	Behavior
Platform Management Console: **Symphony Workload > Configure Applications** Open the application profile and edit the **Session Type Definition** section.	Preemption Rank = *n*	• Specifies the preemption rank, a numerical value from 1 -10000. • The default preemption rank is the lowest possible value,1. • To help protect an important session from losing a resource, specify a higher rank for the session. When there are multiple resources that could be reclaimed, the system may reclaim resources used by lower-ranking sessions first. • If you do not enable selective reclaim, setting the preemption rank may not have a significant effect.

Configuration to modify preemption criteria

By default, the preemption criteria depends on the scheduling policy, and minimizes system overhead.

If you do not have selective reclaim enabled, changing the preemption criteria may not have a significant effect on reclaim behavior.

You can modify this behavior so that the system reclaims resources from recently started tasks first.

Configuration source	Setting	Behavior
Platform Management Console: **Symphony Workload > Configure Applications** Open the application profile and edit the **General Settings** section.	Preemption Criteria = MostRecentTask	• From all the candidate sessions, find the task with the least run time, and reclaim the resource it is using. • If multiple tasks have the same run time, choose one at random. • If multiple tasks run on a slot, consider the cumulative run time of all tasks on that slot.
	Preemption Criteria = Default and Scheduling Policy = Proportional Scheduling or Minimum Services	• (Default setting) Reclaim resources from the most over-allocated session first. • If multiple sessions are equally over-allocated, pick one at random. • If no sessions are over-allocated, pick the least under-allocated.
	Preemption Criteria = Default and Scheduling Policy = Priority Scheduling	• (Default setting) If multiple resources are available, pick one at random.

Resource reclaim interface

Actions to monitor

You can monitor resource reclaim through the Platform Management Console.

Platform Management Console option	Description
Resources > Monitor Resource Allocation	- Displays a list of consumers along with each consumer's current allocation of owned, shared, and borrowed slots and the consumer's current demand

Actions to control

Once you have configured borrowing, lending, and sharing for your cluster, you cannot directly control or release reclaimed resources. When you modify the resource plan and click Apply, changes take effect immediately and could trigger resource reclaim.

User	Interface	Behavior
- Cluster administrator (EGO)	From the command line: `egosh resource close -reclaim` *resource_name*	- Closes a resource, preventing further allocation. The system reclaims the host before it closes; running workload units are re-queued after the configured grace period.
- Application developer	Using the API: onServiceInterrupt	- Notifies the service that the service instance manager has sent an interrupt signal.

Actions to display configuration

User	Command	Behavior
- Cluster administrator - Consumer administrator	From the Platform Management Console: - **Consumers > Consumers & Plans > *consumer_name* > Consumer Properties > Reclaim behavior**	- Displays the settings for **Reclaim grace period** and **Rebalance when resource plan changes or time interval changes**
- Cluster administrator - Consumer administrator	From the Platform Management Console: - **Consumers > Consumers & Plans > Resource Plan> Show Advanced Settings > Expand All**	- Displays the ownership, rank, lend, borrow, and share settings for all consumers
- Cluster administrator	From the Platform Management Console: - **Cluster > Summary > Cluster Properties > Specify resource allocation behavior**	- Displays the settings for **Reclaim shared resources** and **Reclaim lent resources before borrowing**
- Cluster administrator - Consumer administrator	From the Platform Management Console Dashboard: - **Symphony Workload > Monitor Workload > Application Properties** From the command line: - `soamview app` *app_name* `-l`	- Displays the setting for **Selective Reclaim**

User	Command	Behavior
- Cluster administrator - Consumer administrator	From the Platform Management Console Dashboard: - **Symphony Workload > Monitor Workload > Application Properties** From the command line: - `soamview app` *app_name* `-l`	- Displays the setting for **Preemption Criteria**
- Cluster administrator - Consumer administrator	From the Platform Management Console Dashboard: - **Symphony Workload > Monitor Workload > *application_name* > *Session ID* > Session Properties** From the command line: - `soamview session` *application_name:session_ID* `-l`	- Displays the setting for **Preemption Rank**

Feature: Adjustable Share to Workload for Resource Allocation/Reclaim

Adjustable share to workload affects how resources are allocated and reclaimed from consumers.

Use this feature if you want share ratio to always be honored when two or more consumers are competing for resources. Whenever consumers compete for resources, resources are reclaimed and distributed in proportion to share ratio. Workload among all consumers is taken into account and resource distribution is adjusted according to workload. Consumers can reclaim more resources than their configured planned share ratio.

When this feature is not enabled (default behavior), resources are distributed according to configured share ratio but resource distribution is not adjusted according to workload. When two consumers compete for resources, consumers that have not received up to their share ratio can only reclaim resources up to their share ratio. When consumers have reached their share ratio, distribution of additional resources to consumers is done in First-Come, First-Served order.

Resource Reclaim and Distribution Behavior

Reclaim behavior(default and feature enabled)

	Default behavior (feature not enabled)	Feature enabled
When to reclaim	• If a consumer is not using up to its share and now has new demand, the consumer reclaims slots from other consumers who borrowed from the share pool.	• Same behavior as with feature not enabled.
From which consumer to reclaim	• Configured share ratio is a minimum guarantee.	• Configured share ratio is a mininum guarantee.
	• Share quota (the number of slots a consumer can possibly get) is always fixed, equal to the configured share for the consumer.	• Share quota (the number of slots a consumer can possibly get) is dynamic, it is the configured share plus any slots not used by other consumers. When other consumers do not have demand, the share quota of consumers with demand increases.
	• Slots are reclaimed from any consumers who borrowed. If a consumer is not using more than its share ratio, no slots can be reclaimed from the consumer.	• Slots are reclaimed only from consumers who borrowed more than their share.
	• A consumer cannot reclaim slots from other consumers who borrow.	•

	Default behavior (feature not enabled)	Feature enabled
	• From which consumer to reclaim first depends on the consumer tree --- if the parent is using more than its share, slots from the child consumer are more likely to be reclaimed than from another consumer whose parent is not using more than its share.	• From which consumer to reclaim first depends on the share ratio across the entrire consumer tree. The consumer that has borrowed the most when evaluated against its share ratio is reclaimed first. To identify the consumer who borrowed the most against its share ratio, the system evaluates from top to bottom of the consumer tree. Parents are evaluated first. For example, when comparing two parents and parent A has borrrowed more slots than parent B according to its share ratio, the system then evaluates the children of parent A to identify the child which has borrowed the most according to its share ratio. The system will then compare the children of parent A against each other to see from which one to reclaim slots.
	If two parent consumers are using more than their share, resources are reclaimed from the lowest priority parent first.	

Distribution behavior(default and feature enabled)

Default behavior (feature not enabled)	Feature enabled
• When a consumer has demand, the consumer is allocated slots up to its share. The number of slots a consumer can borrow is limited by its share.	• When a consumer has demand, the consumer is allocated slots up to its share. If it has more demand, the consumer can additionally borrow slots from the share pool that are not used by other consumers. The number of slots a consumer can borrow depends on cluster workload. Consumers without demand are not included in calculation of share quota. Slots are distributed among consumers with demand according to their share ratio.
• When a consumer does not have demand, its share goes to the share pool. Slots in the share pool can be borrowed by any consumer, in First-Come, First-Served order.	• When a consumer does not have demand, its share goes to the share pool. Slots in the share pool can be borrowed by consumers with demand according to their share ratio.

Default behavior (feature not enabled)	Feature enabled
• If two consumers request resources at the same time, the highest priority consumer will get all slots in the share pool if it has enough demand. If two consumers request resources at the same time and the highest priority consumer does not have enough demand to get all slots in the share pool, the next consumer with the higher priority will get whichever number of slots is left over.	• If two consumers request resources at the same time, consumers are allocated slots according to their share ratio.
• Whenever a consumer borrows from the share pool, the borrowed resources can be reclaimed if other consumers have demand only if other consumers are not using their share.	• Whenever a consumer borrows from the share pool, the borrowed resources can be reclaimed according to share ratio if other consumers have demand, even if the consumer is not using its share.

Examples: How resource reclaim works when not based on share ratio (feature not enabled)

The default behavior for resource reclaim is to reclaim resources according to consumer priority. Resources are reclaimed from the lowest priority consumer. Share ratio is not taken into account for resource reclaim.

Scenario: Three Consumers, 18 slots

Total share pool: 18 slots

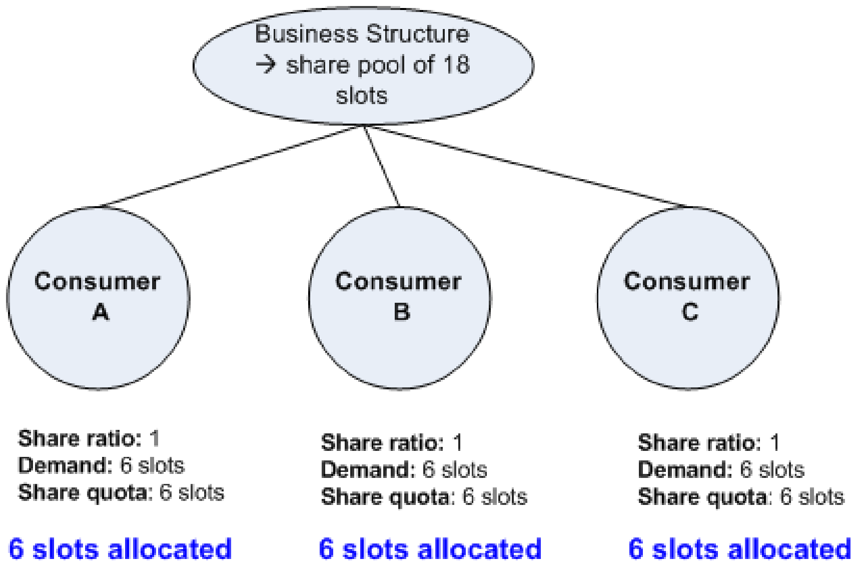
There are three consumers: A, B, C. There are no subconsumers.

The lowest priority consumer is consumer B.

The share ratio is equal for all consumers (1:1:1).

Example: Demand exists for all consumers

All consumers have demand for 6 slots, so the total share pool is distributed among all consumers. Each consumer is allocated 6 slots.



Example: No demand for consumer C

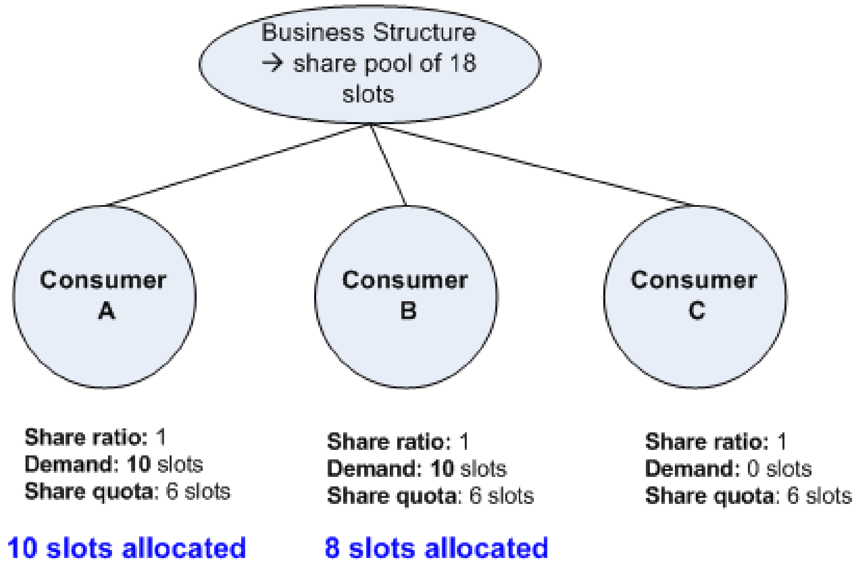
Building on the previous example, all consumers are allocated 6 slots.

Consumer A now has demand for 10 slots.

Consumer B now has demand for 10 slots.

Consumer C no longer has demand.

Because Consumer C does not have demand, its 6 slots can now be allocated between the other consumers. Consumer A is allocated 4 additional slots, the lower priority consumer B, 2 additional slots.



Example: Demand exists again for Consumer C

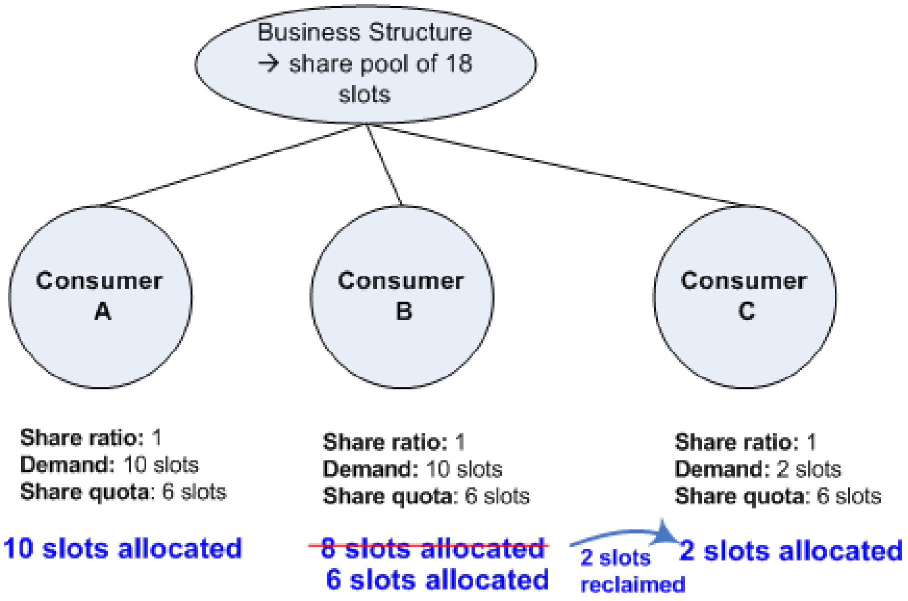
Building on the previous example:

Consumer A has demand for 10 slots, allocated 10 slots.

Consumer B has demand for 10 slots, allocated 8 slots.

Consumer C now has demand for 2 slots.

To determine from which consumer slots should be reclaimed, the system looks at the lowest priority consumer that is allocated more than its share (Consumer B), and reclaims 2 slots to allocate to Consumer C.



How resource reclaim works based on share ratio (feature enabled)

When resource reclaim based on share ratio is enabled, resources are reclaimed according to share ratio across the entire consumer tree.

Scenario: Three Consumers, 18 slots

Let us use the same example used for when the feature is not enabled.

Total share pool: 18 slots

There are three consumers: A, B, C. There are no subconsumers.

The lowest priority consumer is consumer B.

The share ratio is equal for all consumers (1:1:1).

Example 1: Demand exists for all consumers

All consumers have demand for 6 slots.

Behavior is exactly the same as when resource reclaim based on share ratio is not enabled.

Example 2: No demand for consumer C

Behavior is exactly the same as when resource reclaim based on share ratio is not enabled.

Example 3: Demand exists again for consumer C

Building on the previous example:

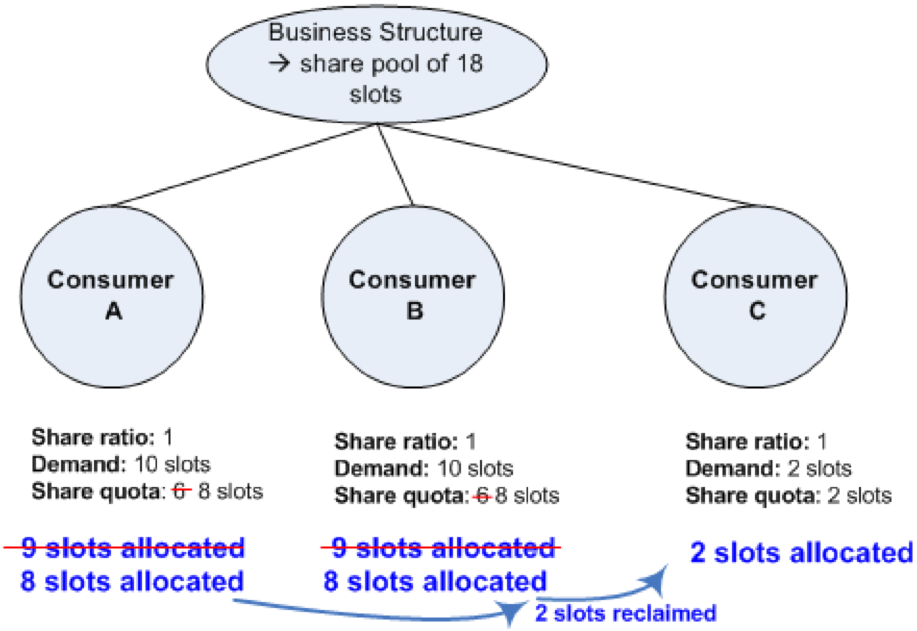
Consumer A has demand for 10 slots, allocated 9 slots.

Consumer B has demand for 10 slots, allocated 9 slots.

Consumer C now has demand for 2 slots.

To determine from which consumer slots should be reclaimed, the system looks at the share ratio across the entire consumer tree. The share ratio is 1:1:1.

So, to distribute slots according to share ratio, the sytem identifies which consumer is the most overallocated. In this case, consumer A is overallocated by 1 slot and consumer B is also overallocated by 1 slot. As a result, the system reclaims 1 slot from each consumer (A and B) to allocate to consumer C.



Scenario: Subconsumers, 100 slots

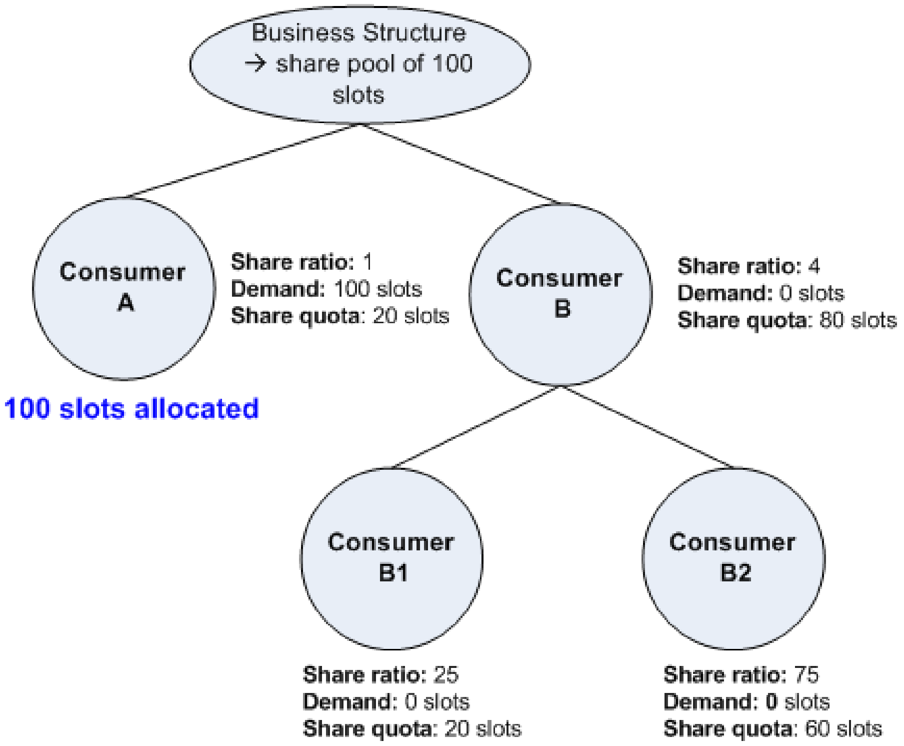
Total share pool: 100 slots

There are two top-level consumers: Consumer A and Consumer B. The share ratio is Consumer A: 1, Consumer B: 4

In Consumer B, we have leaf consumers B1 and B2. The share ratio is: Consumer B1:25, Consumer B2: 75.

Example 1: Only consumer A has demand

Consumer A has demand for 100 slots. Only consumer A has demand, so the total share pool is distributed 100% to Consumer A.



Example 2: B1 has demand for 500 slots

Building on the previous example, consumer A is now allocated 100 slots.

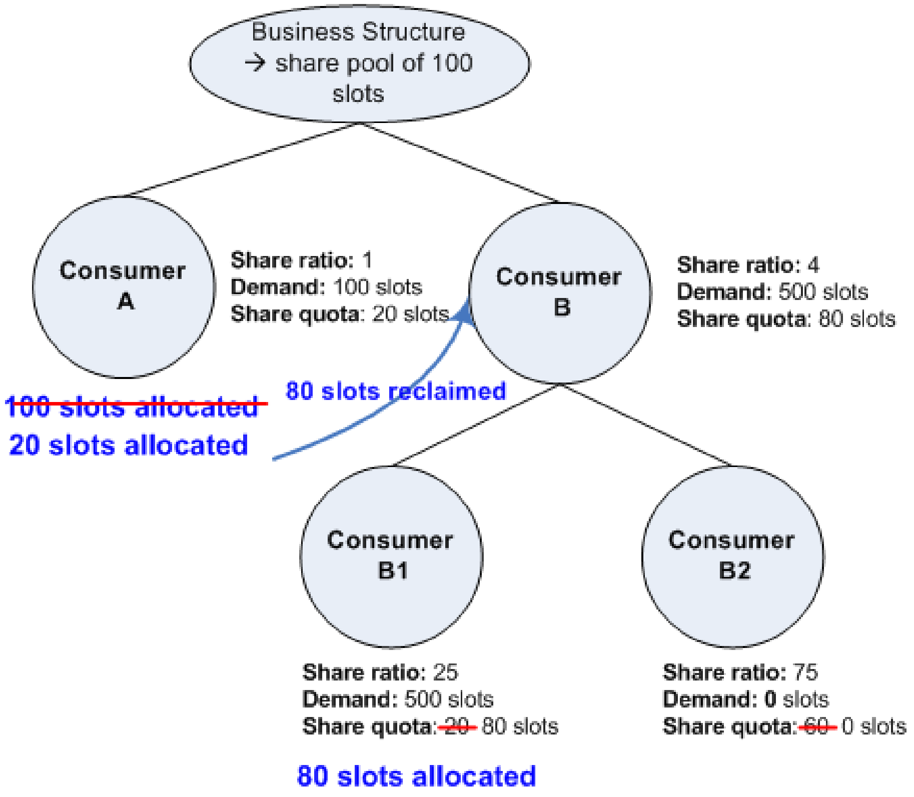
B1 now has demand for 500 slots.

Since consumer B1 has demand, consumer B is considered to have demand by the system.

The system considers the share ratio across the entire tree to reclaim slots. Consumer A’s ratio is 1, consumer B’s ratio is 4, so out of 100 slots, consumer A should get 20, consumer B, 80.

80 slots are reclaimed from consumer A and allocated to consumer B. Consumer B is a top-level consumer, so its leaf consumers are considered.

Consumer B1 has demand. Consumer B2 has no demand. As a result, consumer B1 gets 100% of the slots (80 slots).



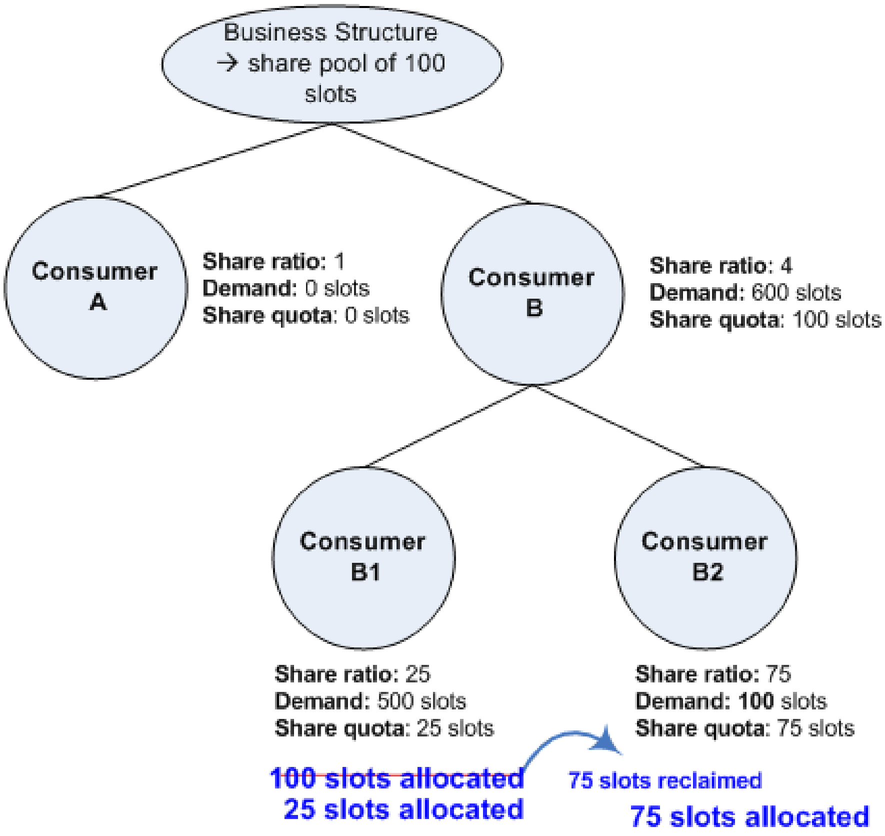
Example 3: All consumers have demand for slots, B2 has demand for 100 slots

Building on the previous example, consumer A now has no demand. Consumer B1 has demand for 500 and is allocated 100 slots.

B2 now has demand for 100 slots.

As a result, slots need to be reclaimed. The system evaluates the share ratio of the entire consumer tree. There is demand across all consumers.

Consumer B1’s ratio is 25, Consumer B2’s ratio is 75. As a result, consumer B1 should get 25, consumer B2 75. Consumer B1 is currently allocated 100 slots. As a result, 75 slots are reclaimed from B1 and allocated to B2.



Configuration to enable adjustable share ratio to workload

Configuration source	Setting	Behavior
ego.conf	EGO_ADJUST_SHARE_TO_WORKLOAD =y	When set to y, specifies that share ratio is to always be honored when two or more consumers are competing for resources. Whenever consumers compete for resources, resources are reclaimed and distributed in proportion to share ratio. Workload among all consumers is taken into account and resource distribution is adjusted according to workload. When set to n or undefined, resources are distributed according to configured share ratio but resource distribution/reclaim is not adjusted according to workload. When two consumers compete for resources, consumers that have not received up to their share ratio can only reclaim resources up to their share ratio. When consumers have reached their share ratio, distribution of additional resources to consumers is done in First-Come, First-Served order.

Interface

Actions to monitor

You can monitor resource allocation and reclaim through the Platform Management Console or the command-line interface.

Interface	Description
Platform Management Console: **Dashboard, Symphony Workload**	- Guaranteed Compute=ownership + share quota
Platform Management Console: Dashboard > Consumers>Monitor Resource Allocation	View how many slots are guaranteed for the consumer. This value will change according to whether Adjustable share ratio to workload is enabled or not.
egosh consumer view *consumer_name*	The Share Quota value changes according to configuration. When Adjustable share ratio to workload is not enabled: - Share quota is the number of slots the consumer is guaranteed. It is calculated from the configured planned share ratio and total number of available slots When Adjustable share ratio to workload is enabled: - Share quota is the number of slots the consumer is guaranteed. It is calculated from the configured planned share ratio, total number of available slots, and current workload in the cluster.

Feature: Using Static Share Quota to Minimize Reclaims Across Multiple Resource Groups

This feature works in conjunction with dynamic share pool scheduling and is used to minimize the number of reclaims that can occur when consumers are overusing resources from multiple resource groups.

Scope

Applicability	Details
Operating system	• All host types supported by the Symphony system
Exclusions	• Does not apply to Symphony DE, which does not have resource lending, borrowing, and reclamation

How dynamic share pool scheduling works

Dynamic share pool scheduling is defined as the scheduling policy in effect when EGO_ADJUST_SHARE_TO_WORKLOAD is enabled in the `ego.conf` file; see *Adjustable Share to Workload for Resource Allocation/Reclaim* on page 220 for details. To better understand how static share quota can be used to minimize the number of reclaims, let’s review how dynamic share pool scheduling works.

With dynamic share pool scheduling, if only one consumer has demand, it deserves 100% of the share pool. If there are multiple resource groups, EGO tries to allocate resources from one resource group after another, which means that a consumer will use all the resources up to its "dynamic share quota" from the first resource group before it gets resources from the next one. Dynamic share quota takes into account the static share ratio and the workload that consumers are running in the resource group at that moment. For example:

- 3 consumers : A, B, C
- static share ratio: 1:2:3
- 1 resource group (RG1) with a total of 18 slots

If only A has workload, its dynamic share quota is 18 slots. If A and B have workload, the dynamic share quota for each consumer will be A=6 slots and B= 12 slots.

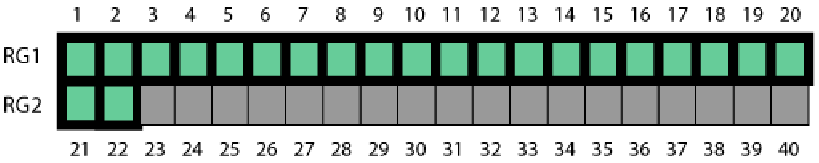
If the first resource group is completely used up when a second consumer has demand, that consumer will use free slots in the remaining resource groups. If EGO cannot satisfy the second consumer’s entire demand after trying to allocate resources from all of the resource groups, the dynamic share quota for each resource group is applied, which can result in unnecessary reclaims. In this case, since the first consumer is overusing resources in the first resource group, its resources will be reclaimed because the second consumer still has unsatisfied demand.

Here is an example of dynamic share pool scheduling with the static share quota feature disabled.

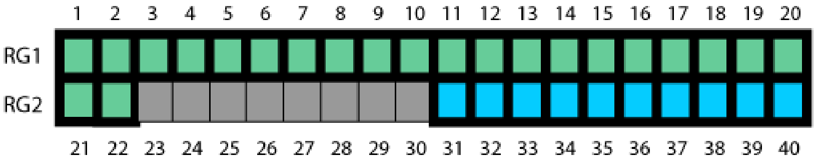
Configuration:

- Resource group RG1 : 20 slots
- Resource group RG2 : 20 slots
- Consumer A : share ratio 1; resource group: RG1, RG2
- Consumer B : share ratio 1; resource group: RG1, RG2
- Consumer C : share ratio 1; resource group: RG1, RG2

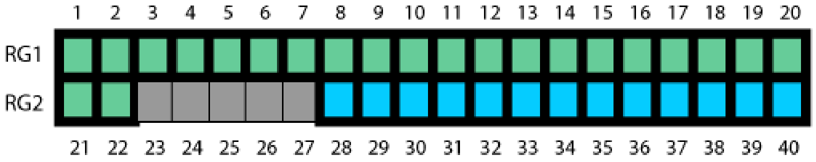
t0: 22 tasks are submitted from consumer A
Consumer A gets all of its dynamic share quota, i.e., 20 slots from RG1, and then gets 2 additional slots from RG2.



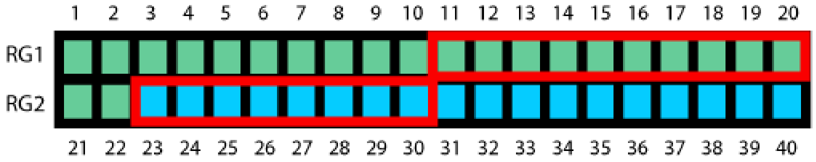
t1: 10 tasks are submitted from consumer B
Consumer B uses the free slots in RG2 first.



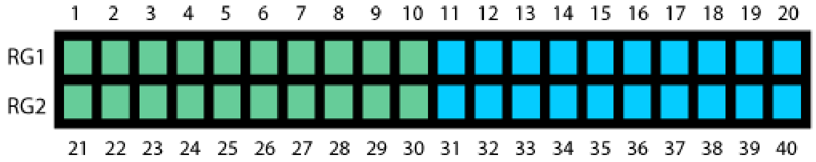
t2: 3 tasks are submitted from consumer B
Consumer B continues to use the free slots in RG2, still within its dynamic share quota for RG2.



t3: 150 tasks are submitted from consumer B
Consumer B gets the remaining slots in RG2. Since consumer B still has unsatisfied demand and there are no free slots left in both resource groups, consumer B goes back to RG1 to reclaim its dynamic share quota.



t4: After reclaim grace period
After consumer B reclaims slots from consumer A, consumer A has unsatisfied demand and starts to reclaim its dynamic share quota from consumer B in RG2. Eventually consumers A and B use their dynamic share quota in both resource groups. As a result, a total of 18 slots have been reclaimed.



About this feature

Applying static share quota to dynamic share pool scheduling

When this feature is enabled, EGO first attempts to allocate resources from each resource group according to the consumer’s "static share quota". The static share quota is derived from the static share ratio for the consumer and the total number of slots in the resource group. For example:

- 3 consumers: A, B, C
- static share ratio: 1:2:3
- 1 resource group (RG1) with 18 slots

Given the preceding configuration, the static share quota of A=3 slots, B=6 slots, and C=9 slots in RG1.

If EGO cannot allocate the consumer’s static share quota of resources because there are not enough idle slots in a resource group, EGO tries to allocate slots from another resource group.

If a consumer has unsatisfied demand after being allocated its static share quota for each resource group, EGO allocates additional resources up to the consumer’s dynamic share quota starting with the first resource group. If the consumer still has unsatisfied demand after all the resources are allocated from the first resource group, EGO allocates resources from the second resource group, and so on. (The order of resource groups is defined by the order that they appear in ConsumerTrees.xml.) As a result, resources used to satisfy the dynamic share quota will not be taken evenly from each resource group.

How resources are reclaimed when the feature is enabled

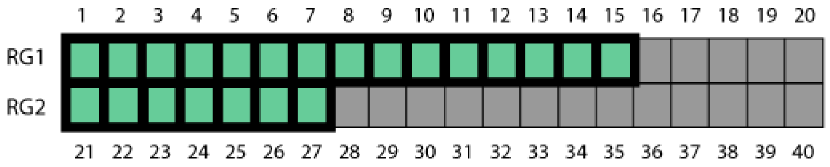
As long as a consumer is consuming within its static share quota for the resource group, the consumer will never be considered to be overusing resources. EGO considers the consumer’s dynamic share quota first when deciding if a consumer is overusing resources.

Reclaiming resources is performed in the following order. In each case, EGO reclaims resources from consumers that are exceeding their dynamic share quota.

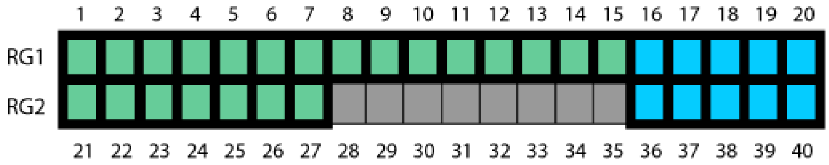
1. If a consumer is not able to get its static share quota from idle slots
2. If a consumer is not able to get its dynamic share quota from idle slots

Using the earlier example of three consumers and two resource groups, the following shows how slots are allocated and reclaimed when the feature is enabled.

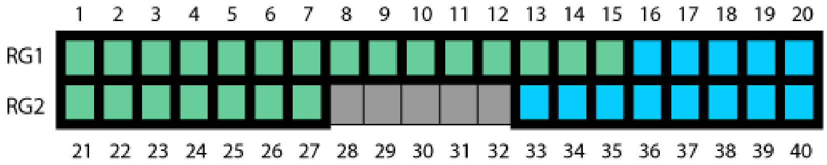
t0: 22 tasks are submitted from consumer A
Consumer A uses its static share quota for each resource group first, i.e., 7 slots from RG1 and 7 slots from RG2. Consumer A then uses the additional slots within its dynamic share quota from RG1.



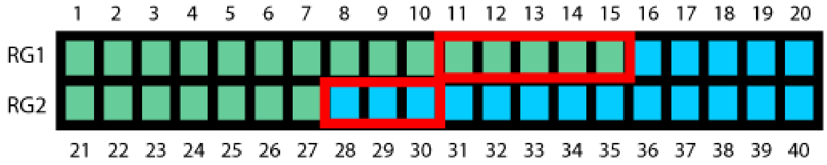
t1: 10 tasks are submitted from consumer B
Consumer B uses all the free slots from RG1(within its static share quota), and then uses additional free slots from RG2.



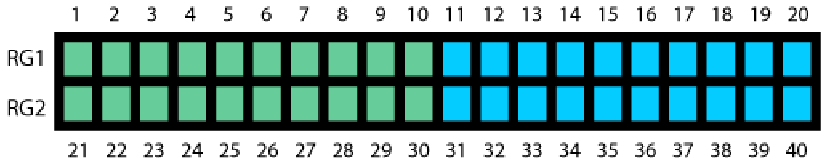
t2: 3 tasks are submitted from consumer B
Consumer B continues to use the free slots in RG2.



t3: 150 tasks are submitted from consumer B
Consumer B uses up the free slots in RG2 but still has unsatisfied demand. Consequently, consumer B reclaims slots from consumer A in RG1 up to its dynamic share quota. After consumer B reclaims from consumer A, consumer A has unsatisfied demand and reclaims slots from consumer B in RG2 up to its dynamic share quota.



t4: After reclaim grace period
After reclaim finishes, consumers A and B get their dynamic share quota of slots. The total number of reclaimed slots is 8. When compared to the previous example, 10 less slots were reclaimed.



As can be seen in the example, since the resource distribution order was done according to the consumers' static share quota in each resource group, the number of reclaimed slots is reduced significantly.

Enabling the feature to minimize reclaims

Configuration source	Setting	Behavior
ego.conf	EGO_ENABLE_BASE_QUOTA=y	When set to y, specifies that resources will be allocated to consumers according to their static share quota from each resource group first, followed by resources allocated according to the order of resource groups defined in ConsumerTrees.xml. The default value is n (not enabled). **Note:** For this feature to work, you must also enable EGO_ADJUST_SHARE_TO_WORKLOAD in ego.conf.

Feature: Using workload preemption to reclaim resources

You can configure a session to be preemptive so that when the session is under-allocated, it can preempt workload of other sessions instead of waiting for other sessions to voluntarily release slots.

The following criteria is used to identify preemption candidates in the given order:

1. Select all over-allocated sessions
2. Order preemption candidates by preemption rank and choose the sessions with lowest rank

   If there are multiple sessions in the lowest rank, choose the session according to the preemptionCriteria configuration, i.e., either a session with the most recently started task or the most over-allocated session (default). In cases where a few tasks must be preempted at the same time, the tasks with the smallest sum of elapsed computation times are chosen.

Session preemption characteristics:

- Preemption can only be triggered by under-allocated sessions.
- An under-allocated session that is configured to be preemptive will preempt over-allocated sessions if the under-allocated session has any unsatisfied demand.
- Workload preemption only happens if the under-allocated session's preemption rank is higher than or equal to the rank of the over-allocated session. Whether a session preempts another session of lower or equal rank or only preempts a session of lower rank is determined by the setting of the preemptionScope attribute in the application profile Sessions cannot preempt other sessions with higher rank.
- Only over-allocated slots will be preempted from a session. Preemption will not cause any session to become under-allocated.
- If multiple slots are shared by more than one session, the session with the highest preemption rank is taken into consideration. In this case, the lower rank session is "protected" by the higher rank session and will not be preempted. Similarly, if multiple sessions on one host share the same slot, the session with the highest preemption rank is taken into consideration.
- If multiple slots are shared by more than one session, the session with the highest preemption rank is taken into consideration. In this case, the lower rank session is "protected" by the higher rank session and will not be preempted. Similarly, if multiple sessions on one host share the same slot, the session with the highest preemption rank is taken into consideration.
- If the SSM cannot find any slots in the lowest rank sessions to preempt, either because these slots cannot be used by the session or these slots are "protected" by higher ranked sessions or under-allocated sessions, the SSM will consider the next higher ranked sessions.
- Preemption takes effect immediately. The service instance manager and service instance are restarted and assigned to under-allocated sessions.

Configuring workload preemption

Preemption scope is configured in the Consumer element of the application profile. The default value for preemptionScope is LowerOrEqualRankedSessions.

If preemptive is set to true for a session type, the under-allocated sessions of this session type can preempt other over-allocated sessions. The default value for preemptive is false.

The preemptionRank attribute defines the session’s rank in relation to other sessions. (Sessions with a lower preemption rank will get preempted before sessions with a higher rank.) The default value for preemptionRank is 1.

Example:

```
<Consumer preemptionScope="LowerRankedSessions"/>
<SessionTypes><Type name="type1" … preemptive="true" preemptionRank="2"/>
```

The preemptionScope, preemptive, and preemptionRank attributes can be configured through the PMC or by manually editing the application profile.

Overriding configured parameters via API

At session creation time, a client can override the session type's preemptive and preemptionRank parameters via the API. Once the session is created, these parameters cannot be changed. Refer to the API Reference in the Knowledge Center for more information.

Best practices

Here are some preemption guidelines for various types of workload. Note that the preemption rank values are provided as an example.

Type of Workload	Preemption Rank
For workload that you do not want preempted, set the preemption rank to the highest level.	20
For normal workload that can be preempted without consequence, set the preemption rank to the lowest level.	10

Feature: Host Scavenging Feature

Host scavenging allows you to leverage the compute power of hosts that would ordinarily not be added to the cluster, like desktop computers and servers. When idle, these resources run work sent from an application manager (such as Platform Symphony).

An additional software license is required to use this feature. This feature is packaged with Symphony and does not require separate deployment.

Contents

- About host scavenging
- Scope
- Configuration to enable host scavenging
- Host scavenging behavior
- Configuration to modify host scavenging
- Host scavenging commands

About host scavenging

The host scavenging feature adds hosts to the cluster that only run work when they are idle. Local users on the hosts are not interrupted, but once they are no longer using the host, the host is used in the cluster. When the host is used again to run non-Symphony applications, the host is closed to the cluster and work runs on other hosts.

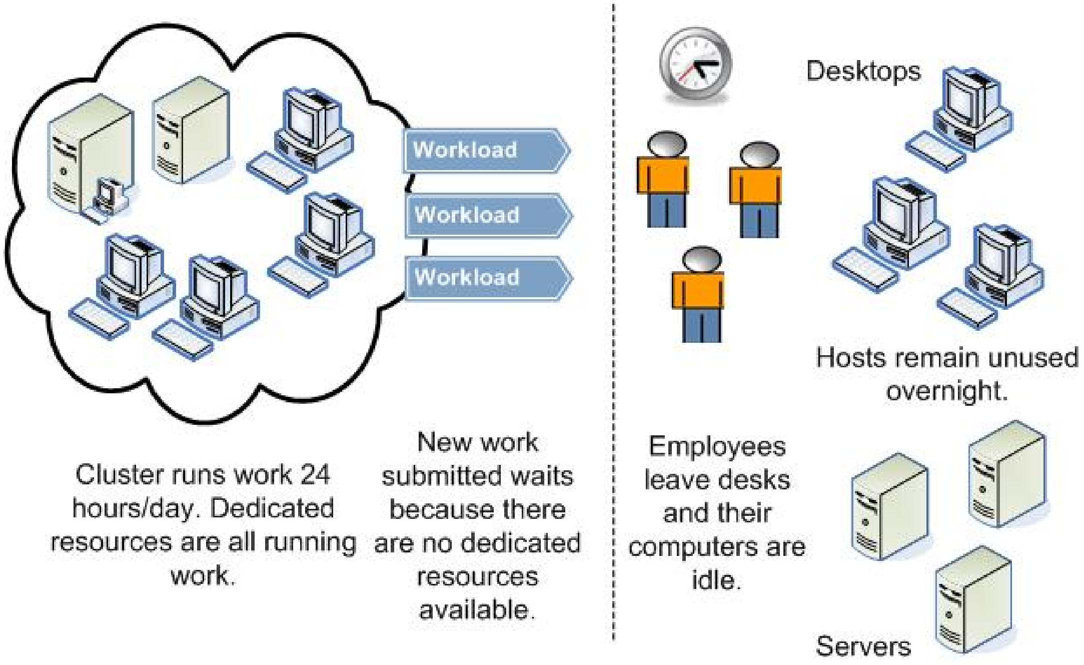

Workload
Workload
Workload
Cluster runs work 24 hours/day. Dedicated resources are all running work.
New work submitted waits because there are no dedicated resources available.
Desktops
Hosts remain unused overnight.
Employees leave desks and their computers are idle.
Servers

Figure 1: Host scavenging not enabled (default)

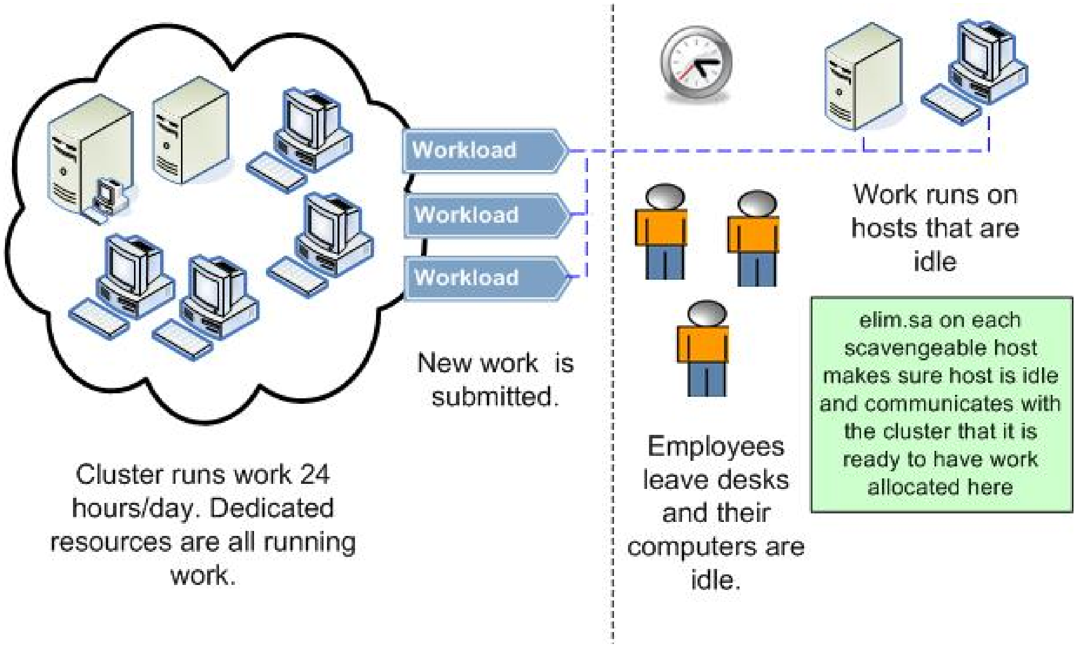

Workload
Workload
Workload
New work is submitted.
Cluster runs work 24 hours/day. Dedicated resources are all running work.
Employees leave desks and their computers are idle.
Work runs on hosts that are idle
elim.sa on each scavengeable host makes sure host is idle and communicates with the cluster that it is ready to have work allocated here

Figure 2: Host scavenging enabled

Summary of host scavenging process

1. A scavenging agent, `elim.sa`, is included in the EGO package and deployed to hosts during installation. This agent collects data used for host scavenging.
2. An administrator enables the host scavenging feature on selected hosts.
3. The scavenging agent monitors the local load information. EGO opens or closes the local host based on the load information collected by the scavenging agent. If the host is closed , it is also reclaimed from the consumer using it. The workload running on the host is terminated and requeued.
4. When EGO closes the scavenged host and the host is reclaimed, it no longer qualifies for allocation to any consumer until it is opened again. The host will be opened, or scavenged, automatically based on the configured thresholds and load information collected by the scavenging agent.

Resource groups for scavenge-ready hosts

As a best practice for the host scavenging feature, you should set up a resource group of scavenge-ready hosts .This separates opportunistic (scavenge-ready) hosts from dedicated hosts used deterministically by the cluster. You set this resource group to exclude management hosts and include hosts with the static resource tag "scvg" for desktop hosts or "svrscvg" for server hosts. Once set up, any new host added to the cluster with either resource tag "scvg" or "svrscvg" automatically joins its respective resource group. A scavenge consumer should be created and configured to own all slots that will be scavenged. Lending should be enabled without limit.

Scope

Applicability	Details
Operating system	• Linux/UNIX • Windows

Applicability	Details
Security	• No security issues
Preconditions	• Platform Symphony must be installed on all hosts participating in scavenging.

Specifying processes that automatically close hosts

There may be situations when you do not want a host to be scavenged if a specific process is running on that host. For example, there may be high-priority non-Symphony application processes that require the host. Symphony allows you to specify these processes in a list will cause the host to be closed if the process is running on that host. When Symphony detects one of these processes are running, it automatically closes the open host or prevents the host from opening for future scavenging. These processes are specified in a list known as the close process list. You can populate this list when you configure host scavenging.

Note:

Since script (.bat or .sh) files generally wrap the true process name within the file, do not define a script file name as a process name. Instead, define the process name that the script file launches. Defining script file names as process names can result in unexpected behavior.

Excluding processes in the calculation of CPU utilization

The scavenging feature allows you to specify processes that will not be taken into account by the scavenging agent in its calculation of CPU utilization. For example, there may be a periodic process like a virus scan program that causes high CPU usage but only for a short time when it runs. For such cases, you can specify the processes that you want to exclude from the CPU utilization calculation. This CPU utilization, referred to as Adjusted CPU Utilization, will be lower than the true CPU utilization of the host if a specified process is running.

The processes to be excluded from the Adjusted CPU utilization can be specified during scavenging configuration.

Note:

Since script (.bat or .sh) files generally wrap the true process name within the file, do not define a script file name as a process name. Instead, define the process name that the script file launches. Defining script file names as process names can result in unexpected behavior.

Symphony-launched workload processes, which could cause the host to toggle between open and closed states, are automatically excluded from the calculation of Adjusted CPU Utilization.

Host scavenging configuration and control

This feature can be configured and enabled using the Platform Management Console. Refer to *Configure and enable desktop scavenging* on page 246 and *Configure server scavenging* on page 248 for detailed steps.

Host scavenging configuration and control using CLI

This feature is enabled by running the following commands.

Scavenge-ready hosts need both the scavenge resource tag and the agent control flag set.

Note:

If you configured the host as scavenge-ready during the Symphony installation on that host, it is not necessary to configure the resource tag.

- Scavenge resource tag (scvg): Marks a desktop host as scavenge-ready and allows it to be identified with a desktop scavenge resource group.
- Scavenge resource tag (svrscvg): Marks a server host as scavenge-ready and allows it to be identified with a server scavenge resource group.
- Agent control (agent_control): Enables or disables EGO's ability to open and close the host using the information collected by the local scavenging agent. The value can be on, fastrelease, or off.

Where	Command	Description
On each desktop host that you want scavenged	`egoconfig addresourceattr "[resource scvg]"`.	Adds a "scvg" tag to desktop hosts to indicate they are scavenge-ready.
On each server host that you want scavenged	`egoconfig addresourceattr "[resource svrscvg]"`.	Adds a "svrscvg" tag to server hosts to indicate they are scavenge-ready.
From any host	`egosh ego elimrestart SA on` *host_name*	Sets the "agent_control" flag to "on" and enables EGO to scavenge the specified host(s) using default threshold values. When the host is closed, the grace period for workload is honored.
From any host	`egosh ego elimrestart SA fastrelease` *host_name*	Sets the "agent_control" flag to "fastrelease" and enables EGO to scavenge the specified host(s), using default threshold values, but not honoring the grace period for workload when the host is closed.

Follow the steps in *Enable host scavenging using the CLI* on page 250 to set up this feature.

Host scavenging behavior

Scavenge-ready host states and status

When the scavenging agent detects that the host is busy, EGO closes the host. The running workload is terminated after a grace period and the host is prevented from further allocation.

The host status changes to `closed` and the reason indicates that EGO closed the host based on the configured thresholds for load information.

Note that the reclaim grace period set for a consumer does not apply when a scavenge-ready host is configured using the fastrelease command option.

When a scavenged host starts and stops running cluster work

EGO opens a host when the following conditions, in the given order, are met to indicate that a host is not busy and ready for opportunistic workload.

Criteria	Description	Preconditions for triggering scavenging
1. Close processes	These processes cannot be running if the host is to be opened or remain open.	None of the processes specified in the close process list is running
2. User idle time threshold (applicable to desktop scavenging only)	User idle time of the host in minutes	User idle time threshold is exceeded
3.a. CPU idle time threshold	CPU idle time threshold of the host in minutes. The CPU idle time takes into account the Adjusted CPU Utilization threshold. As soon as the Adjusted CPU Utilization is below its threshold, the CPU idle time count begins. If, at any time, the Adjusted CPU Utilization is above its threshold, the CPU idle time is reset. When the CPU idle time reaches its threshold and the other conditions are also met, the host is opened.	CPU idle time setting is exceeded
3.b. Adjusted CPU Utilization threshold	Adjusted CPU utilization of the host does not include CPU utilization caused by processes specified in the exempt process list. Adjusted CPU Utilization is a factor in the determination of CPU idle time and is expressed as a percentage	Adjusted CPU Utilization is lower than its threshold

A desktop scavenged host is closed when it starts to be used locally, as determined by the user idle time being below its threshold setting. EGO closes the host and the host is reclaimed. CPU idle time is not a factor when EGO closes a desktop scavenged host. For scavenged server hosts, CPU idle time is considered and when it goes below its threshold setting, the host is closed and reclaimed.

For both desktop and server hosts, once the scavenge trigger conditions are met again (indicating that the host is not busy once more), the host is automatically opened.

Defaults

When no thresholds are specified, the following default values are used.

Threshold	Default Value
User idle time (minutes)	10
Adjusted CPU Utilization (%)	0
CPU idle time (minutes)	10

Configuration to modify host scavenging using the CLI

Modify host scavenging in the following ways:

- Configuration to define thresholds
- Configuration to define process lists
- Configuration to disable host scavenging
- Configuration to disable the grace period

- Configuration to change process priority

Configuration to define thresholds

Command	Example	Behavior
`egosh ego elimrestart SA on,` uit_t,cu_t,cit_t *host_name* ..	`egosh ego elimrestart SA on,2,0.3,1.67 host1`	Changes the threshold values for host1 to • User idle time threshold of 2 minutes (For server scavenging, user idle time (uit_t) must be set to 0.) • Adjusted CPU utilization threshold of 30% • CPU idle time threshold of 1.67 minutes (or 100 seconds)

You can modify the default threshold values that determine when EGO opens and closes the scavenged host.

Configuration to define process lists

Command	Example	Behavior
`egosh ego elimrestart SA on` -p *exempt_processes host_name* ...	`egosh ego elimrestart SA on -p cmd.exe host1`	The scavenging agent monitors the processes running on the host. If a process that is defined as an exempt process is running, the scavenging agent excludes its CPU utilization from its total CPU utilization calculation.
`egosh ego elimrestart SA on` -c *close_processes host_name* ...	`egosh ego elimrestart SA on -c cmd.exe host1`	The scavenging agent monitors the processes running on the host. If a process that is defined as a close process is running, EGO automatically closes the hosts. The host remains closed as long as the defined process is running.

Configuration to disable host scavenging

Command	Example	Behavior
`egosh ego elimrestart SA off` *host_name* ...	`egosh ego elimrestart SA off host1`	The scavenging agent continues to monitor the scavenge-ready hosts but EGO no longer opens or closes them according to the thresholds set. Until you delete the scavenge resource group, work can continue to run on these hosts. **Note:** Use the keyword `all` to disable scavenging on all hosts running it at once. Otherwise, the command is for the specified hosts only or if no hosts are specified, only for the local host.
Delete the scavenge resource group and the scavenge consumer.	N/A	Once you delete the scavenge resource group and the scavenge consumer, as long as the hosts do not belong to any other resource groups, work is no longer allocated to those hosts.

Configuration to disable grace period

Command	Example	Behavior
`egosh ego elimrestart SA fastrelease` *host_name* ...	`egosh ego elimrestart SA fastrelease host1`	When a predefined threshold is reached, EGO closes the host and terminates running workload without a grace period.

Host scavenging controls and commands

Configuration and monitoring using the PMC

Command	Description
Scavenging > **Configuration** > **... Scavenging General Setting**	Configure and enable host scavenging for desktop and server hosts.
Resources > **Configure Resource Groups** for scavenge resource group: List of member hosts	All hosts that are listed in the scavenge resource group in the **Member hosts** section and have the state **ok** are scavenge-ready. Add the status column using the table preferences.
Resources > **Monitor/Control Hosts** > **Hosts (List View)**: Scavenging Control	Hosts that are listed with **Scavenging Control** as **on** have the agent control flag turned on, meaning the scavenging agent is monitoring and EGO is controlling the opening and closing of the host according to the threshold values set.
Resources > **Monitor/Control Hosts** > **Hosts (List View)**: Resource Attr	Hosts that are listed with **scvg** or **svrscvg** have the scavenge resource tag applied to them. These hosts are scavenge-ready and are dynamically added to a resource group that specifies a resource requirement of **select (!mg && scvg)** or **select (!mg && svrscvg)**.

Command	Description
Resources > **Monitor/Control Hosts** > **Hosts (List View)**: User Idle Time Threshold, Adjusted CPU Util Threshold, and CPU Idle Time Threshold	View the thresholds set for each scavenge-ready host. Scroll to the right to see the values set for the thresholds for user idle time, Adjusted CPU utilization, and CPU idle time.

Hosts need both the agent control set to on and the scavenge resource tag (scvg or svrscvg) applied for host scavenging to function properly. If a host is missing one of the two, the feature does not work properly.

CLI commands to control and monitor

Command	Description
`egosh ego elimrestart SA on` *host_name* ...| `all`	Turns host scavenging on for a specific host or for all hosts with the resource "scvg" tag associated with them. Uses default threshold values.
`egosh ego elimrestart SA off` *host_name* ...| `all`	Turns host scavenging off for local (if no host specified), a specific host name, or all hosts (using the keyword `all`).
`egosh ego elimrestart SA on, 2,0.3,1.67 host_name` ...| `all`	When turning host scavenging on, you can also set the threshold values. Note that user idle time (uit_t) must be set to 0 for server scavenging.
`egosh ego elimrestart SA fastrelease` host_name...| all	When turning host scavenging on, you can also disable the grace period (fastrelease). By default, the grace period is enabled.
`egosh resource list -o status,ut,it,agent_control,uit_t,cu_t,cit_t` *`host_name`*	Lists the scavenge-related information for a host.

CHAPTER

10

Working with Scavenged Hosts

General workflow for setting up host scavenging

1. Install Platform Symphony on hosts intended for scavenging. During installation choose whether to designate the host as a desktop or server scavenge-ready host
2. If the host was not preconfigured as a scavenge-ready host during Symphony installation, add "scvg" or "svrscvg" tag to the host and start or restart the cluster.
3. Enable the scavenging feature.
4. Create a resource group for scavenge-ready hosts.
5. Create or modify a consumer to include the scavenge resource group.
6. Set ownership, sharing, lending, and borrowing for scavenge resource group.
7. Set thresholds that trigger scavenging. EGO will open and close the hosts using the information collected by the local scavenging agent.
8. Verify the hosts that are scavenge-ready.

Configure and enable desktop scavenging

1. In the Platform Management Console, click Scavenging > Configuration > Desktop Scavenging General Setting.
2. Do one of the following:

   - To modify an existing configuration, click the configuration link.
   - To add a new configuration, in the Global Actions dropdown list, select Add New Desktop Scavenging Configuration.
3. Set the User idle time threshold.

   This setting is a trigger for desktop host scavenging. The user idle time is affected locally by physical mouse or keyboard movements and actions. The user idle time can also be affected by remote connections depending on the local OS platform and the type of remote connection software used.

   The user idle time of the host must be above the threshold setting as one of the preconditions for host scavenging. Another precondition is the CPU idle time.
4. Set the CPU idle time threshold.

   This setting is configured in conjunction with Adjusted CPU Utilization threshold as a trigger for host scavenging. As soon as the Adjusted CPU Utilization is below its threshold, the CPU idle time count begins. If, at any time, the Adjusted CPU Utilization is above its threshold, the CPU idle time is reset. When the CPU idle time reaches its threshold and the other conditions are also met, i.e., no close processes are running and user idle time is above its threshold, the host is opened for scavenging.
5. Set the Adjusted CPU Utilization threshold.

   Adjusted CPU Utilization of a host excludes CPU utilization by user-specified processes (see next step). Adjusted CPU Utilization is a factor in the determination of CPU idle time. The Adjusted CPU Utilization of the host must be below this setting to start the CPU idle time counter. The threshold value is a percentage expressed as a decimal.
6. In the Exempt processes textbox, enter the names of processes on the host that you want to exclude from the calculation of Adjusted CPU Utilization. Separate multiple process names with a comma (,).

   An example of a process you may want to exclude is a periodic process like a virus scan program, which may cause high CPU usage for a short time. Note that CPU utilization caused by Symphony-launched workload processes are automatically excluded from the calculation of Adjusted CPU Utilization.
7. If you want to close a host to Symphony workload when a specific process is running on the host, enter the process name in the host scavenging close processes textbox. Separate multiple process names with a comma (,).
8. Choose whether you want to release the host immediately or after a grace period once the host is closed to scavenging. The grace period is configured on the Consumer Properties page.
9. To apply this configuration to specific hosts:

   a) Filter the available hosts using the Filter host by dropdown list.
   b) Select the appropriate hosts in the list under Available hosts.
   c) Click Add.

      The selected hosts are displayed in the list under Enable scavenging control on hosts with this configuration.
10. Click either Create or Save, whichever is applicable, to apply the scavenging configuration to the selected hosts.

The scavenging configuration is applied to each selected host and the agent control flag is set to "on", which means the scavenging agent is monitoring and EGO is controlling the opening and closing of the host according to the threshold values set.

Configuration example

To help you understand how the scavenging parameters are related, let’s take a look at a configuration example:

- User idle time threshold: 10.0 minutes
- CPU idle time threshold: 5.0 minutes
- Adjusted CPU Utilization threshold: 0.40
- Scavenging host close processes: abc.exe

In this example, the host will be open for scavenging if process `abc.exe` is not running on the host, the user idle time is above 10 minutes, and Adjusted CPU Utilization is below 0.40 (40%) for a duration of 5.0 minutes.

Configure server scavenging

1. In the Platform Management Console, click Scavenging > Configuration > Server Scavenging General Setting.
2. Do one of the following:
   - To modify an existing configuration, click the configuration link.
   - To add a new configuration, in the Global Actions dropdown list, select Add new server scavenging configuration.
3. Set the CPU idle time threshold.

   This setting is configured in conjunction with Adjusted CPU Utilization threshold as a trigger for host scavenging. As soon as the Adjusted CPU Utilization is below its threshold, the CPU idle time count begins. If, at any time, the Adjusted CPU Utilization is above its threshold, the CPU idle time is reset. When the CPU idle time reaches its threshold and the other conditions are also met, i.e., no close processes are running and user idle time is above its threshold, the host is opened for scavenging.
4. Set the Adjusted CPU Utilization threshold.

   Adjusted CPU Utilization of a host excludes CPU utilization by user-specified processes (see next step). Adjusted CPU Utilization is a factor in the determination of CPU idle time. The Adjusted CPU Utilization of the host must be below this setting to start the CPU idle time counter. The threshold value is a percentage expressed as a decimal.
5. In the Exempt processes textbox, enter the names of processes on the host that you want to exclude from the calculation of Adjusted CPU Utilization. Separate multiple process names with a comma (,).

   An example of a process you may want to exclude is a periodic process like a virus scan program, which may cause high CPU usage for a short time. Note that CPU utilization caused by Symphony-launched workload processes are automatically excluded from the calculation of Adjusted CPU Utilization.
6. If you want to close a host to Symphony workload when a specific process is running on the host, enter the process name in the host scavenging close processes textbox. Separate multiple process names with a comma (,).
7. Choose whether you want to release the host immediately or after a grace period once the host is closed to scavenging. The grace period is configured on the Consumer Properties page.
8. To apply this configuration to specific hosts:
   a) Filter the available hosts using the Filter host by dropdown list.
   b) Select the appropriate hosts in the list under Available hosts.
   c) Click Add.

      The selected hosts are displayed in the list under Enable scavenging control on hosts with this configuration.
9. Click either Create or Save, whichever is applicable, to apply the scavenging configuration to the selected hosts.

   The scavenging configuration is applied to each selected host and the agent control flag is set to "on", which means the scavenging agent is monitoring and EGO is controlling the opening and closing of the host according to the threshold values set.

Configuration example

To help you understand how the scavenging parameters are related, let’s take a look at a configuration example:

- CPU idle time threshold: 5.0 minutes
- Adjusted CPU Utilization threshold: 0.40
- Scavenging host close processes: abc.exe

In this example, the host will be open for scavenging if process `abc.exe` is not running on the host and Adjusted CPU Utilization is below 0.40 (40%) for a duration of 5.0 minutes.

Enable host scavenging using the CLI

Platform Symphony must be installed and running.

1. On each host that you want to scavenge, ,run `egoconfig addresourceattr "[resource scvg]"` (for desktop scavenging) or`egoconfig addresourceattr "[resource svrscvg]"` (for server scavenging) before joining the host to the cluster. Note that this step is not necessary if the host was configured as a scavenge-ready host during Symphony installation.

   Adds "scvg" or "svrscvg" resource tag to hosts to indicate they are scavenge-ready.
2. Restart EGO on the hosts you added the resource tag to.
3. From any host, run `egosh ego elimrestart SA on` *host_name* ....

   Sets the agent control flag to "on" and enables EGO to scavenge the specified hosts. The following default threshold values are used:

   - Idle time threshold in minutes: 10 (Note: For server scavenging, you must set the idle time to 0)
   - Adjusted CPU utilization threshold as a percentage: 0
   - CPU idle time threshold in minutes: 10

   The thresholds are configurable.

   Note:

   The remaining steps are optional, but recommended.
4. From the Platform Management Console, create a new dynamic resource group for scavenge-ready hosts.

   a) Name the group in such a way that you can identify it as a group of scavenge-ready hosts.
   b) Select Hosts filtered by resource requirement and in the resource requirement, specify: `select (!mg && scvg)` (for desktop scavenging) or `select (!mg && svrscvg)` (for server scavenging)

   Note:

   If the scavenge resource group is not the last resource group appearing in the resource plan, the resources from the scavenge resource group might be used before resources from other resource groups. If you want to ensure that the dedicated grid resources are used before the scavenge-ready resources, you should create the scavenge resource group last. This way, work is only allocated to your scavenge-ready resources if all other resources are busy. This principle only applies if you complete the remaining steps and no other resource groups are added after the scavenge group.

   The resource group dynamically includes any host that is not a management host and has the "scvg" resource tag associated with it.
5. In all other resource groups that use compute hosts, add

   ```
   !scvg
   ```

   or

   ```
   !svrscvg
   ```

   , as applicable, to the resource requirement.

   This prevents host overlaps between resource groups.
6. From the Platform Management Console, identify the applications that you want to have work running on scavenged hosts.

   a) Modify the consumer properties to include the new scavenge resource group for each application that should run on scavenged hosts.
   b) Check that the grace period specified is appropriate. Work is terminated on a scavenged host when it becomes busy, taking the grace period into consideration. A lower grace period terminates the work faster than a higher grace period.

Note:

Specify any number of resource groups in addition to the scavenge resource group if you want the work to run on dedicated cluster hosts as well. Work only runs on scavenged hosts if dedicated hosts are not available. Again, this principle is only true if you complete the optional steps in this procedure.

7. Create a new consumer and name it so you can identify that it is used only for scavenging.
   a) Specify the new scavenge resource group as the consumer's only resource group.

   This consumer owns all the slots of the scavenge-ready resource group and lends them to other consumers as needed.

Note:

Never register any applications to this consumer.

8. Modify the resource plan for the scavenge resource group.
   a) Specify the scavenge consumer to own all slots that will be scavenged.

      You can set the scavenge consumer to own more slots than are displayed if you know more scavenge-ready host slots will be available in the future.
   b) Enable lending but specify no limit.
   c) Leave borrowing disabled.
   d) Disable sharing.

      This introduces a non-editable default setting.
9. Enable borrowing in the resource plan for the consumers associated with applications that you want to make use of scavenge-ready hosts.

Stop workload from running on a scavenged host

You must be a cluster administrator to enable host scavenging.

Restriction:

Host scavenging is a feature for use with Platform Symphony.

There are two ways to stop workload from running on a scavenged host:

- A cluster administrator closes a host and the resource is reclaimed. Any running workload is re-queued after a grace period. New workload cannot be assigned. Slots become unavailable for allocation by EGO.
- EGO closes the host after a predefined threshold is reached. If fastrelease is enabled, any running workload is re-queued without a grace period. If fastrelease is not enabled, any running workload is re-queued after a grace period. By default, fastrelease is not enabled. Once EGO closes the host, new workload cannot be assigned. Slots become unavailable for allocation by EGO.

Note:

If the SSM fails to re-queue the running workload and return the host within the defined grace period, then EGO forcefully terminates the SIs and re-queues the workload activity before reclaiming the host.

1. Navigate to Resources > Monitor/Control Hosts > Hosts (List View).
2. For the host you wish to close, click Actions and then select Close.

   The Close host dialog displays..
3. Select Close and reclaim this host.... Click Apply.
4. Remove the hosts from any scavenge resource groups.

Verify the hosts currently available for scavenging

For planning or monitoring purposes, you may wish to identify at a glance those hosts that are scavenge-ready (configured) and those that are currently available to participate in scavenging (enabled). This is normally done through the Platform Management Console.

For host scavenging to work properly, hosts must be both configured and enabled for host scavenging.

Restriction:

Host scavenging is a feature for use with Platform Symphony.

1. From the Console, click Resources > Monitor/Control Hosts > Hosts (List View).

   Note:

   You cannot view scavenge-ready or enabled hosts from **Hosts (Icon View)**.

2. If Resource Attr does not display as a column heading in the resource list, click Preferences, and then select Resource Attr from the checklist.
   a) From the checklist, also select Scavenging Control (agent control flag).
   b) Set the sort order to either Resource Attr (to easily identify those hosts in the list configured for scavenging) or Scavenging Control (to easily identify those hosts in the list with scavenging enabled).
   c) Click Apply to set the preferences to the host list, and then click Close.
3. Locate the Resource Attr column in the host list to see which hosts are tagged (configured) as scavenge ready.

   Those hosts that can be potentially scavenged display the resource tag scvg or svrscvg.

4. Locate the Scavenging Control column in the host list to see which hosts have scavenging turned on or off.
   - On: Indicates that this host is currently available (enabled) to participate in host scavenging and to run opportunistic workload assigned by Platform Symphony.
   - Off: Indicates that EGO is not upholding the thresholds by opening or closing the host.

Set trigger conditions (thresholds) for host scavenging using the CLI

You must be a cluster administrator. You must be logged on to Windows as the local systems OS account administrator or logged on to Linux/UNIX as the root OS account.

Indicate thresholds using the command line. The scavenging agent (SA) uses the thresholds to evaluate trigger conditions that enable EGO to open the host for scavenging.

Restriction:

Host scavenging is a feature for use with Platform Symphony.

1. Run
   ```
   egosh ego elimrestart SA on,
   <user_idle_time_threshold_in_minutes>,<Adjusted_CPU_utilization_threshold_in_percen
   tage>,<CPU_idle_time_threshold_in_minutes> host_name
   ```
   ...

   For example:

   ```
   egosh ego elimrestart SA on,2,0.3,1.7 Host1
   ```

   This example enables (turns “on”) desktop scavenging on Host1, sets the user idle time threshold (`uit_t`) to 2 minutes, the Adjusted CPU utilization threshold (`cu_t`) to 30%, and the CPU idle time threshold (`cit_t`) to 1.7 minutes (100 seconds).

   Note:

   To enable and configure server scavenging, the user idle time threshold must be set to 0.

2. Run
   ```
   egosh resource list -o agent_control,uit_t,cu_t,cit_t Host1
   ```
   (no spaces before or after commas) to ensure the agent control flag is turned on and that the threshold indices are set as intended.

FAQs for host scavenging

In Windows, what actions trigger the closing of desktop hosts and reclaiming of work during host scavenging?

The user idle time is only affected by local physical mouse or keyboard movements/actions. The movement of a remote client’s or terminal server client's mouse/keyboard does NOT affect the user idle time.

Does EGO allocate slots to a consumer from a scavenge resource group first or last?

During processing, EGO considers the order in which resource groups are stored in `ConsumerTrees.xml`. The order is based on when the resource group was created (that is, the first resource group created gets processed first). If you want to ensure that scavenge-ready hosts are processed last (for example, slots are not allocated from scavenge-ready hosts in a scavenge resource group until hosts from all other resource groups are first considered by EGO), do not create a scavenge resource group until you have created all other resource groups.

If the scavenge resource group is not the last resource group appearing in the resource plan, the resources from the scavenge resource group might be used before resources from other resource groups. Assuming that when you create your resource groups, you created the scavenge group last, the processing order is as follows:

	Resource Group A (created first)	**Resource Group B (created second)**	**Resource Group C, made up of scavenge-ready hosts (created last)**
All guaranteed allocations processed first	1st --->	2nd --->	3rd --->
All reclaim requests processed second	4th --->	5th --->	6th --->
All borrowing requests processed last	7th --->	8th --->	9th (last)

Following this recommended processing order, EGO allocates slots to a consumer from a scavenge resource group after processing other resource groups first. Likewise, EGO processes reclaim requests and borrows hosts from a scavenge resource group last.

If you created a scavenge resource group during installation without considering the creation order, you can manually edit `ConsumerTrees.xml` and re-order the groups. There is some risk involved in editing this file, so it is best to contact Platform Technical Support for assistance.

PART

III

Advanced Cluster Configuration

CHAPTER

11

Installation

Parallel installation

You have installed a second cluster to run in parallel with an original cluster, but you have not started the second cluster yet.

If you have installed one cluster and want to install a second cluster in parallel with the first as a means of transitioning from the old cluster to the new cluster, you must make some changes after installing but before starting the second cluster.

1. Change the service start type from automatic to manual in `named.xml` located in `$EGO_CONFDIR/../../eservice/esc/conf/services`.

   ```
   <sc:StartType>Manual</sc:StartType>
   ```
2. Comment out the dependency on Service Director in `wsg.xml` located in `$EGO_CONFDIR/../../eservice/esc/conf/services`.

   ```
   <!-- <sc:Dependency type="OnStart">ServiceDirector</sc:Dependency>
   ```
3. Start the second cluster.
4. When the first cluster has been decomissioned, stop the WebServiceGateway service.
5. Change the start type back to automatic in `named.xml` and remove the comment on service director dependency in `wsg.xml`.
6. Start the WebServiceGateway service.

Executing commands in a multi-cluster environment

Symphony users can run EGO commands from one host that may connect with one or more clusters and have this command take effect in a specific cluster.

Note:

All clusters must be the same version, for example, Symphony 3.2.

Applies only to Symphony users.

Install and set your environment to run egosh commands in a multi-cluster environment

Whether you are an interactive user or not, follow these steps to be able to run `egosh` commands in a multi-cluster environment.

You must complete these steps each time you want to be able to run `egosh` on a different cluster.

1. On the host that you want to run in a multi-cluster environment, install the Symphony client package.
2. Copy the folder `%EGO_TOP%\1.2.4u1\bin` from the master host to this host.
3. Set the environment variable PATH in the `symclientenv.bat` file under `SymphonyClient\conf` on the host to include the directory containing the `egosh` binary (`%EGO_TOP%\1.2.4u1\bin\egosh.exe`).
    a) If the PATH variable is not set, modify `symclientenv.bat` file so that it points %EGO_CONFDIR% of the Symphony shared directory of the cluster to be connected to. For example: `\\FileServer\SymShare\kernel\conf`.
    b) In `ego.conf` under `SymphonyClient\conf`, modify it by adding the master list and VEMKD port number.
4. Run `symclientenv.bat`.
5. Run `egosh resource list`.

    A list of resources displays if the configuration was successful.

You can now issue `egosh` subcommands from this host to another cluster.

CHAPTER

12

Ports, TCP connections, and IPV6 support

Summary of Ports Used by Symphony

Ports used by Symphony are defined with the base port at installation.

The default base connection port is 7869. Symphony always uses seven consecutive ports starting from the base port at installation. By default, Symphony uses ports 7869-7875.

For the base port, both TCP and UDP are required. For other ports, only TCP is required.

Component	Default port used	Need to enable in		Notes	Where to configure
		TCP	UDP		

lim	7869	Y	Y	Base connection port.	Windows:
					`%EGO_CONFDIR%\ego.conf, EGO_LIM_PORT parameter`
					Linux/UNIX:
					`$EGO_CONFDIR/ego.conf, EGO_LIM_PORT parameter`
					You can also set the base port with the command egoconfig setbaseport. This command also sets the vemkd, pem, egosc, session director, repository service ports from the base port.

vemkd
(Resource Manager)
	7870	Y	N	Automatically assigned from the base port at installation.	Windows:
					`%EGO_CONFDIR%\ego.conf, EGO_KD_PORT parameter`
					Linux/UNIX:
					`$EGO_CONFDIR/ego.conf, EGO_KD_PORT parameter`

pem	7871	Y	N	Automatically assigned from the base port at installation.	Windows:
					`%EGO_CONFDIR%\ego.conf, EGO_PEM_PORT parameter`
					Linux/UNIX:
					`$EGO_CONFDIR/ego.conf, EGO_PEM_PORT parameter`

egosc (EGO Service Controller)	7872	Y	N	Automatically assigned from the base port at installation.	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\egosc_conf.xml, ESC_PORT` parameter Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/egosc_conf.xml, ESC_PORT parameter`

Component	Default port used	Need to enable in		Notes	Where to configure
		TCP	UDP		
rs (Repository Service)	7873	Y	N	Automatically assigned from the base port at installation.	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\rs.xml`, REPOSITORY_SERVICE_PORT parameter Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/rs.xml, REPOSITORY_SERVICE_PORT parameter`
sd (Session Director)	7874 7875	Y	N	Requires two ports. Automatically assigned from the base port at installation.	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml`, SD_ADMIN_PORT and SD_SDK_PORT parameters Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml, SD_ADMIN_PORT and SD_SDK_PORT parameters`
ssm (Session Manager)	Any	Y	N	No fixed port numbers by default. Ports are automatically assigned from available ports in the operating system. Ports are reassigned whenever the Session Manager is restarted.	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml`, SSM_SDK_ADDR and SSM_SIM_ADDR parameters Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml, SSM_SDK_ADDR and SSM_SIM_ADDR parameters`

Component	Default port used	Need to enable in		Notes	Where to configure
		TCP	UDP		

Web server	8080
	8005
	8009
	8443
		Y	N	The first port is the client connection port. You must know this port to connect to the Platform Management Console.
				Ports 8005 and 8009 are for administration.
				Port 8443 may be required when SSL is enabled. This may be optional as the default port for SSL is 8080.
					Windows:
					`%EGO_CONFDIR%\..\..\gui\conf\server.xml`
					Linux/UNIX:
					`$EGO_CONFDIR%/../../gui/conf/server.xml`

Web service gateway	9090	Y	N	Since the web service gateway might run on any management host in the cluster, the web service gateway port must be free on all management hosts.	Windows: %EGO_CONFDIR%\wsg.conf, WSG_PORT parameter Linux/UNIX: `$EGO_CONFDIR/wsg.conf`, WSG_PORT parameter
Service director	53	Y	N	Symphony requires exclusive use of port 53 on the service director. Since the service director might run on any management host in the cluster, port 53 must be free on all management hosts.	This value cannot be changed.

Derby database	1527	Y	N		Windows:
					`%EGO_CONFDIR%\..\..\perf\conf\datasource.xml`
					`%EGO_CONFDIR%..\..\eservice\esc\conf\services\derby_service.xml`
					Linux/UNIX: `$EGO_CONFDIR/../../perf/conf/datasource.xml`
					`$EGO_CONFDIR/../../eservice/esc/conf/services/derby_service.xml`

Component	Default port used	Need to enable in		Notes	Where to configure
		TCP	UDP		

plc
(Loader Controller)
	4046	Y	N		Windows:
					`%EGO_CONFDIR%\..\..\perf\conf\plc.xml`
					Linux/UNIX:
					`$EGO_CONFDIR/../../perf/conf/plc.xml`

Symphony client	Any	Y	N	Used with firewalls on the client side and rfa command.	EGO_CLIENT_ADDR=*port_number* or *port_range* Example: EGO_CLIENT_ADDR=56000-56020
				Used with direct data transfer feature. You can define a port or port range for the client to listen for connections from the service. You may want to do this if your client is running behind a firewall. If the SOAM_DIRECT_DATA_PORT is not defined, Symphony will use the value defined in EGO_CLIENT_ADDR.	SOAM_DIRECT_DATA_PORT=*port_number* or *port_range*

Connection ports and base port

On every host, a set of connection ports must be free for use by EGO components.

Symphony requires exclusive use of certain ports for communication. Symphony uses the same seven consecutive ports on every host in the cluster. The first of these is called the base port.

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

The ports can be customized by customizing the base port. For example, if the base port is 6880, Symphony uses ports 6880-6886.

Symphony needs the same ports on every host, so you must specify the same base port on every host.

To change the base port after installation, shut down the cluster and use the `egoconfig setbaseport` command on each host in the cluster. Start the cluster to use the new connection ports.

Web server ports

On the web server, a set of ports must be free.

EGO requires exclusive use of three communication ports on the web server. By default, EGO uses ports 8080, 8005, and 8009.

The first port is the client connection port. You must know this port to connect to the Platform Management Console.

The other ports are for administration.

Since the web server might run on any management host in the cluster, the web server ports must be free on all management hosts.

The ports can be customized by editing `Connector port` in the server configuration files.

Windows:

- `%EGO_CONFDIR%\..\..\gui\conf\server.xml`

Linux/UNIX:

- `$EGO_CONFDIR/../../gui/conf/server.xml`

Web service gateway port

EGO requires exclusive use of a communication port on the web service gateway. By default, EGO uses port 9090.

Since the web service gateway might run on any management host in the cluster, the web service gateway port must be free on all management hosts.

This port can be customized by editing the `wsg_port` value in the web service gateway configuration file:

- Windows: %EGO_CONFDIR%\wsg.conf
- Linux/UNIX: $EGO_CONFDIR/wsg.conf

Service director port

EGO requires exclusive use of port 53 on the service director. This value cannot be changed.

Since the service director might run on any management host in the cluster, port 53 must be free on all management hosts.

Service controller port

Port 7872 is automatically assigned to the EGO service controller during installation. This port can be customized by editing the ESC_PORT value in the service controller configuration file.

- Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\egosc_conf.xml`
- Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/egosc_conf.xml`

Ports used by Symphony

In addition to the ports used by EGO, Symphony uses several ports for session manager and the session director.

Ports used by EGO and Symphony are defined with the base port at installation.

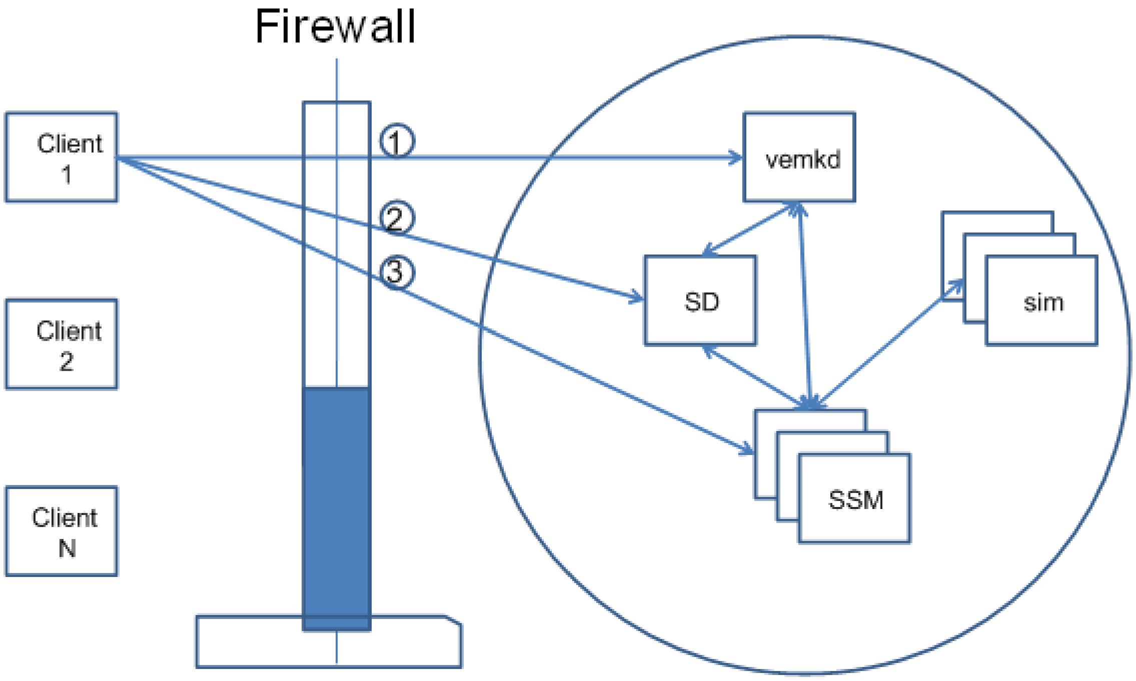
Configuration files are usually located in the eservice directory under the directory in which Symphony was installed. If you have set share for management hosts, they are under the shared directory.

Component	Default port	Configuration file
Repository Service	7873	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\rs.xml` Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/rs.xml`
Session Director	7874 7875	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml` Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml`
Session Manager	Any, by default	Ports are automatically assigned from available ports in the operating system. Ports are reassigned whenever the Session Manager is restarted. Configuration file: Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml` Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml`

Firewall support

Client firewalls

This topic is only applicable to the grid version of Symphony.



You need to open up a few ports (vemkd, Session Director, session manager) in the firewall for the client to interact with the grid. The client first communicates with the vemkd and gets the port information of the Session Director. The client then connects to the Session Director and gets the port information of the session manager and then connects to the session manager. It is important to set appropriate port ranges for session managers because there may be multiple session managers in a Symphony grid.

By default, vemkd and Session Director are configured with fixed port numbers. It is required to configure the session manager ports for firewall support in the `sd.xml` file:

Component	Configuration file
vemkd (Resource Manager)	Windows: `%EGO_CONFDIR%\ego.conf` Linux/UNIX: `$EGO_CONFDIR/ego.conf`
Session Director	Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml` Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml`

Component	Configuration file
Session Manager	It is required to configure the Session Manager ports for firewall support: Windows: `%EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml` Linux/UNIX: `$EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml`

In addition to port configuration, it may be necessary to configure TCP Keep Alive parameters for the connection endpoints in the Symphony environment. The Keep Alive messages are sent through the firewall periodically in order to prevent the firewall from dropping the connection during periods of no user-activity. For the Keep Alive feature to work under realistic conditions, it must be configured to start sending the messages before a routing device's idle time out is triggered. For example, if a firewall is configured to discard idle connections after 15 minutes, you would want your Keep Alive messages to be sent after 10 minutes of inactivity. For more information about TCP Keep Alive configuration, refer to *Configuration of TCP connections* on page 282

Configure

This section describes port configuration when implementing a firewall.

1. To configure a firewall for Symphony, you have to open the ports on the firewall that are required for the connection protocols enabled on your client. Make sure you plan to open the relevant SD, vemkd, and SSM ports.
2. Log on to the master host as the cluster administrator and shut down the Symphony grid completely.
   a) Disable all applications

      ```
      soamcontrol app disable all
      ```
   b) Stop all services

      ```
      egosh service stop all
      ```
   c) Shut down the Symphony grid

      ```
      egosh ego shutdown all
      ```
3. Open the `sd.xml` file and add the port range for the SSM_SDK_ADDR and SSM_SIM_ADDR parameters. Note that you only need to define the SSM_SIM_ADDR parameter if you have a firewall between the SSMs and the SIMs.

   If there is more than one platform, then you need to add the port range for all of them.

   Example:

   ```
   <sc:ActivityDescription>
   ..............................
   <ego:EnvironmentVariable name="SD_ADMIN_PORT">@ADMIN_PORT@</
   ego:EnvironmentVariable><ego:EnvironmentVariable name="SD_SDK_PORT">@SDK_PORT@</
   ego:EnvironmentVariable>
    <ego:EnvironmentVariable name="SSM_SDK_ADDR">31000-32000</
   ego:EnvironmentVariable>
    <ego:EnvironmentVariable name="SSM_SIM_ADDR">32001-33000</
   ego:EnvironmentVariable>
   ................................
   </sc:ActivityDescription>
   ```

Note:

.You should ensure that the port range is sufficient for the number of SSMs that you expect to run on the same host.

Note:

Port range should be equal to or greater than the maximum number of slots in the management host.

4. Start up the Symphony grid.

   ```
   egosh ego start all
   ```

   Enable all the disabled applications.

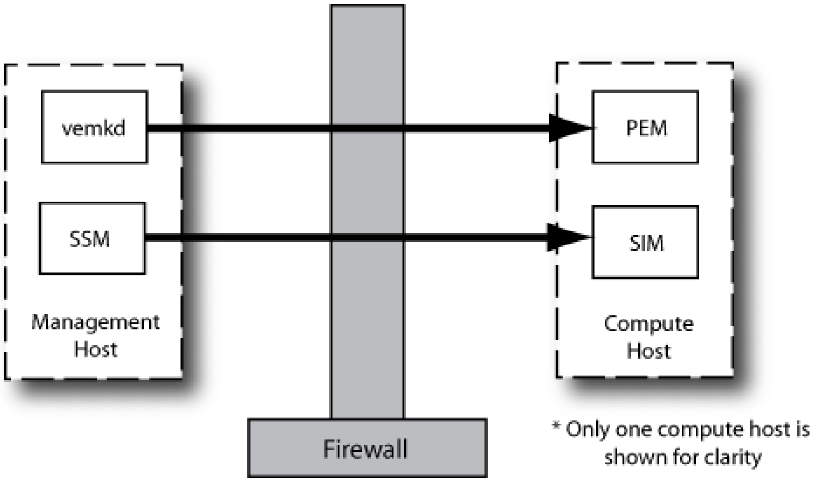
Management and compute host firewalls

Scope

Applicability	Details
Operating system (management hosts)	Windows 2003 and 2008, Linux

Feature details

Symphony offers firewall support to prevent the connection between vemkd on the management host and PEM on the compute hosts from being disconnected by a firewall when the connection remains idle for long periods of time. The same firewall support is also available for the connection between the SSM on the management host and the SIMs on the compute hosts. Firewall support is enabled by configuring the EGO_ENABLE_CHAN_KEEPALIVE parameter.



The feature works by periodically passing TCP Keep-Alive Messages between the management hosts and compute hosts. The interval timing for sending the TCP keep-alive packet while the connection between the management hosts and compute hosts is idle is configurable using the EGO_CHAN_KEEPALIVE_TIME (for connection between vemkd and PEM) and PLATCOMMDRV_TCP_KEEPALIVE_TIME (for connection between SSM and SIM).

Configuration

1. Enable the Symphony ports in the firewall.
2. Enable firewall support between management hosts and compute hosts by configurng EGO_ENABLE_CHAN_KEEPALIVE=y in `ego.conf`. By default, firewall support is disabled.
3. Configure the TCP Keep Alive time for the connection between vemkd and PEM by setting the EGO_CHAN_KEEPALIVE_TIME parameter in `ego.conf`. The setting should be less than the firewall's configured timeout for terminating a connection.The minimum valid value is 180 seconds.
4. Configure the TCP Keep Alive time for the connection between SSM and SIM by setting the PLATCOMMDRV_TCP_KEEPALIVE_TIME environment variable in the SSM section of the application profile. For example:

```
<env name=" PLATCOMMDRV_TCP_KEEPALIVE_TIME ">3600</env>
The setting should be less than the firewall’s configured timeout for terminating a
connection.
```

Change the base connection port

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

The ports can be customized by customizing the base port. For example, if the base port is 6880, Symphony uses ports 6880-6886.

Symphony needs the same ports on every host, so you must specify the same base port on every host.

1. Shut down the cluster.
2. Use the command `egoconfig setbaseport` on each host in the cluster.

   This command also sets the vemkd, pem, egosc, repository service, session director ports from the base port.
3. Start the cluster to use the new connection ports.

Change the session director port

You must be a cluster administrator to perform this task.

1. Log on to the master host in the cluster.
2. Linux/UNIX only. Set the environment. For example, if you installed Symphony in /opt/ego:
   - For `csh` or `tcsh`, use `cshrc.platform`:

     source /opt/ego/cshrc.platform
   - For `sh`, `ksh`, or `bash`, use `profile.platform`:

     . /opt/ego/profile.platform
3. Stop the service:

   egosh service stop SD
4. Open the `sd.xml` configuration file.
5. Change the port numbers specified for the SD_ADMIN_PORT and SD_SDK_PORT parameters in the file.

   The SD_ADMIN_PORT is used for the command line interface and client requests. The SD_SDK_PORT is used by the client and service APIs in your client application or service program.

   For example:

   ```
   <ego:EnvironmentVariable name="SD_ADMIN_PORT">7875</ego:EnvironmentVariable>
   <ego:EnvironmentVariable name="SD_SDK_PORT">7874</ego:EnvironmentVariable>
   ```
6. Save the file.
7. Restart EGO on the master host:

   egosh ego restart

Change the session manager port

You must be a cluster administrator to perform this task.

1. Log on to the master host in the cluster.
2. Linux/UNIX only. Set the environment. For example, if you installed Symphony in /opt/ego:
    - For `csh` or `tcsh`, use `cshrc.platform`:

      source /opt/ego/cshrc.platform
    - For `sh`, `ksh`, or `bash`, use `profile.platform`:

      . /opt/ego/profile.platform
3. Stop the service:

   egosh service stop SD
4. Open the `sd.xml` configuration file.
5. Change the port numbers specified for the SSM_SDK_ADDR and SSM_SIM_ADDR parameters in the file.

   The SSM_SDK_ADDR is used for the command line interface and client requests. The SSM_SIM_ADDR is used by the service instance manager to contact session manager.

   For example:

   ```
   <ego:EnvironmentVariable name="SSM_SDK_ADDR">31000-32000</ego:EnvironmentVariable>
   <ego:EnvironmentVariable name="SSM_SIM_ADDR">32001-33000</ego:EnvironmentVariable>
   ```

   Note:

   Port range should be equal to or greater than the maximum number of slots in the management host.
6. Save the file.
7. Restart EGO on the master host:

   egosh ego restart

Change the repository service port

You must be a cluster administrator to perform this task.

The repository service is the service that handles deployment.

1. Log on to the master host in the cluster.
2. Linux/UNIX only. Set the environment. For example, if you installed Symphony in `/opt/ego`:
   - For `csh` or `tcsh`, use `cshrc.platform`:

     source /opt/ego/cshrc.platform
   - For `sh`, `ksh`, or `bash`, use `profile.platform`:

     /opt/ego/profile.platform
3. Stop the service:

   egosh service stop RS
4. Open the `rs.xml` configuration file.
5. Change the port number specified for the REPOSITORY_SERVICE_PORT parameter in the file. For example:

   ```
    <ego:EnvironmentVariable name="REPOSITORY_SERVICE_PORT">7875</
   ego:EnvironmentVariable>
   ```
6. Save the file.
7. Restart EGO on the master host:

   egosh ego restart

Configuration of TCP connections

This topic is only applicable on Symphony grid.

TCP connection attributes are configured on a cluster basis to optimize data throughput over network connections. These attributes are set in the `sd.xml` configuration file.

The attributes should be configured for each connection endpoint in the Symphony environment, i.e., the client, Session Director, and session manager. The following table lists the relevant attributes.

Tip:

If you want to configure attributes with default values, it is not necessary to add them to the sd.xml file.

Attribute	Default	Notes
TCP_NODELAY	1	May be set to 0 for message aggregation.
TCP_KEEP_ALIVE_TIME	Value derived from current OS setting	OS default is 7200 seconds for most operating systems but may vary.
TCP_SEND_BUFFER_SIZE	65535	Any new value that is less than or equal to the default value is ignored.
TCP_RECV_BUFFER_SIZE	65535	Any new value that is less than or equal to the default value is ignored.

Note:

On Solaris platforms, TCP_KEEP_ALIVE_TIME can only be set system-wide and not on a per socket basis (this is an OS limitation). Symphony on Solaris ignores the TCP_KEEP_ALIVE_TIME option if it is set.

Sample configuration in sd.xml:

```
<ego:EnvironmentVariable name="SDK_TRANSPORT_OPT">TCP_NODELAY=0,TCP_KEEP_ALIVE_TIME=300,
TCP_SEND_BUFFER_SIZE=65536,TCP_RECV_BUFFER_SIZE=65536</ego:EnvironmentVariable>
         ...
<ego:EnvironmentVariable name="SD_SDK_TRANSPORT_OPT">TCP_NODELAY=0,TCP_KEEP_ALIVE_TIME=300,
TCP_SEND_BUFFER_SIZE=65536,TCP_RECV_BUFFER_SIZE=65536</ego:EnvironmentVariable>
         ...
<ego:EnvironmentVariable name="SSM_SDK_TRANSPORT_OPT">TCP_NODELAY=0,TCP_KEEP_ALIVE_TIME=300,
TCP_SEND_BUFFER_SIZE=65536,TCP_RECV_BUFFER_SIZE=65536</ego:EnvironmentVariable>
         ...
```

Configuring local connections on clients

There may be situations where global TCP connection attributes are not appropriate for all connection endpoints in the system; for example, remote clients that are geographically distant from the cluster may require more time to send messages over the network. It is possible to override the system-wide attributes on the remote client host by setting environment variables in the OS shell before starting the client process. This method of overriding system-wide attributes can also be applied to remote hosts that are running services.

The environment variables correspond to the four TCP connection attributes described previously:

Environment variable on client	Overrides this attribute in SD.xml
PLATCOMMDRV_TCP_NODELAY	TCP_NODELAY
PLATCOMMDRV_TCP_KEEPALIVE_TIME	TCP_KEEP_ALIVE_TIME
PLATCOMMDRV_TCP_SEND_BUFFER_SIZE	TCP_SEND_BUFFER_SIZE
PLATCOMMDRV_TCP_RECV_BUFFER_SIZE	TCP_RECV_BUFFER_SIZE

Once the environment variables have been created, you can adjust their values to suit the network environment.

IPv6 support

EGO supports IPv6. Among other benefits of IPv6, longer address lengths (128 bit) result in increased address availability for networked devices, allowing you to allocate addresses in large blocks.

EGO and IPv6

Notes on EGO support for IPv6 include the following:

- Scope: You must assign global unicast addresses to IPv6-enabled hosts.
- Compatibility: IPv4 and IPv6 hosts can only communicate with corresponding hosts of the same type or with dual-stack hosts. However, dual-stack hosts can communicate with both IPv4 and IPv6 hosts, allowing compatibility between the two host types. Therefore, if you have a mixed cluster with both IPv4 and IPv6 hosts, configure it to be dual-stack.
- DNS cache: To locally store host names and mapped addresses (both IPv4 and IPv6 types), EGO uses a host DNS cache.
- Hosts file: You can define a `hosts` file to provide address-to-name translation. Create `hosts` in `$EGO_CONFDIR` (Linux/UNIX) or `%EGO_CONFDIR%` (Windows). This file is most useful if you have multi-homed hosts (hosts with multiple interface cards/nics) or dual-stack hosts. It assists you in authenticating hosts, and viewing host mappings in cases where there are multiple addresses and names.

Configuring for IPv6 support

To enable, disable, and/or configure IPv6 support, set these parameters.

Parameter	**Variables**	**Default**	**File name and location**
EGO_DHCP_ENV	• Undefined (client's IP addresses are cached) • Defined (client's IP addresses are not cached; enables dynamic IP addressing for all client hosts in cluster) If defined, you must also define EGO_DYNAMIC_HOST_WAIT_TIME for hosts to rejoin a cluster after their IP address changes.	Undefined	• Linux/UNIX: *`$EGO_CONFDIR/`* `ego.conf` • Windows: `%EGO_CONFDIR%\ego.conf`
EGO_DUALSTACK_PREFER_IPV6	• Y (returns IPv6 at front) • N (returns IPv4 at front) Meaningful for dual-stack hosts. If set, a dual-stack host uses IPv6 instead of IPv4 to communicate with other IPv6 or dual-stack hosts. (See note.)	N (IPv4 at front)	• Linux/UNIX: `$EGO_CONFDIR/ego.conf` • Windows: `%EGO_CONFDIR%\ego.conf`

Parameter	Variables	Default	File name and location
EGO_ENABLE_SUPPORT_IPV6	• Y (to enable) • N (to disable) Enabling support for IPv6 does not have any effect IPv4-only hosts. If set to N, IPv6-only hosts are not recognized.	N (disabled)	• Linux/UNIX: `$EGO_CONFDIR/ego.conf` • Windows: `%EGO_CONFDIR%\ego.conf`
EGO_HOST_CACHE_DISABLE	• Y (to disable) • N (to enable) Enabling support for IPv6 does not have any effect IPv4-only hosts.	N (enable caching of host names and addresses)	• Linux/UNIX: `$EGO_CONFDIR/ego.conf` • Windows: `%EGO_CONFDIR%\ego.conf`
EGO_HOST_CACHE_NTTL	"Negative time to live", in seconds The amount of time that errors are cached.	20 seconds (to turn off caching completely, set to 0)	• Linux/UNIX: *`$EGO_CONFDIR/`*`ego.conf` • Windows: *`%EGO_CONFDIR%`*`\ego.conf`
EGO_HOST_CACHE_PTTL	"Positive time to live", in seconds The amount of time cached results are stored.	86400 seconds (24 hours) (to turn off caching completely, set to 0)	• Linux/UNIX: `$EGO_CONFDIR/ego.conf` • Windows: `%EGO_CONFDIR%\ego.conf`

Note:

Set `EGO_DUALSTACK_PREFER_IPV6` in `ego.conf` to sort address presentation in the cache library. For example, set EGO_DUALSTACK_PREFER_IPV6 to “**Y**" so that IPv6 addresses appear at the top of the list; set it to “**N**" so that IPv4 addresses appear at the top. This parameter only affects the sort order of the list, and communication between dual-stack hosts. It does not enable IPv6, nor does it filter the address list.

CHAPTER

13

Security

User accounts

Your cluster uses a number of accounts to run properly.

In a mixed cluster, you need one OS account for each platform of the OS accounts.

OS account	Description	Used to ...
Installation OS account	The OS account used when installing on a host.	Not usually used again once installation is complete.
System services execution user OS account	The OS account that runs most of the Platform agents, daemons, and services on a host.	This is the account that runs all the elements of the infrastructure of the cluster, including: • LIM • PEM • EGOSC (EGO service controller) • WEBGUI • Web Service Gateway (WSG) • SD • RS (repository service) • DerbyDB (if applicable) • PLC • Purger • Service Director
Consumer execution user OS account	The OS account that has access and control over workload for individual consumers.	This is the account that runs the following Symphony processes: • SSM • SIM • SI
Application workload execution user OS account	The OS account that runs work. Requires Platform Symphony or another workload component.	Depending on installation and configuration decisions, there could be multiple application workload execution user OS accounts.
Cluster administrator OS account	The OS account that has permission to change cluster configuration and control the cluster.	On Windows management hosts, VEMKD executes with this OS account.

There are also Platform user accounts used in your cluster as well. See more information in the following table.

Platform account	Description	Used to ...
EGO authentication users	An EGO authentication user account is a Platform system user who can be assigned to any role: cluster administrator, consumer administrator, or consumer user. Once you have created user accounts, you can assign roles to them in the Platform Management Console. **Restriction:** Only cluster administrators have access to **User Accounts** in the Platform Management Console.	The Platform user accounts used to log on to the Platform Management Console and the `egosh` command line interface (`egosh user logon`).

Allowable characters

A user account name can have the following characters: any alphanumeric character (a-z, 0-9), dashes (-), and underscores (_).They are case sensitive.

The length limitation of password is 16 and the allowed characters are:

!#$%'()+,-./0123456789;=@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
^_`abcdefghijklmnopqrstuvwxyz{}~

Not supported: &"<>?|*: and characters from languages other than English.

User roles

Symphony, out-of-the-box, has three user roles that can be assigned to any user account. Each user role has a predefined level of accessibility and control in the Platform Management Console.

Role	Description
Cluster Administrator	A "super user" able to accomplish all administrative and workload tasks, with access to all areas of the Platform Management Console and all actions within it. Entry to the Platform Management Console is through the dashboard, a heads up display that gives you a running summary of the health of your cluster.
Consumer Administrator	Access and control only over own branch of the tree. Consumer administrators are assigned at the top-level consumer and they are administrators for all sub-consumers in that branch of the tree. Entry to the Platform Management Console is through the dashboard, a heads up display that gives you a running summary of the health of your cluster.
Consumer User	Access and control over their own workload units only. Consumer users are assigned to individual consumers on your tree.

You can assign one user role to different user accounts for different consumers.

Symphony also offers the ability to create user roles that can be customized to site-specific security requirements; refer to *Customizing user roles* on page 291 for details.

Customizing user roles

Symphony offers administrators the ability to choose from a set of predefined permissions and apply them to new or existing user roles.

Scope

Applicability	**Details**
Operating system	Windows Linux Solaris
Exclusions	Does not apply to Symphony DE since the DE version does not require permissions

About user roles

Symphony, out-of-the-box, allows a system user with an authenticated EGO user account to be assigned any of the following roles: cluster administrator, consumer administrator, or consumer user. Each of these roles are associated with a fixed set of permissions that either grant or deny access to specific system controls and operations. For example, consumer users only have access and control over their own workload units and cannot access workload units of other consumers.

At some sites, cluster administrators/users and their functions may not map exactly to Symphony's preconfigured security model for user roles. It may be desirable to have flexibility in the assignment of privileges. This chapter discusses the permissions available for monitoring and controlling Symphony operations and how to assign them to user roles that you can create.

Permission set

A predefined set of permissions are available for customizing user roles. You can assign any of the following permissions to a role.

Permission	**Description**
standard reports control	Users with this permission are able to view standard reports via the PMC and produce/export standard reports with access to all cluster data.
custom reports control	Users with this permission are able to view custom reports via the PMC and produce/export custom reports with access to all cluster data.
Symphony debug control	Users with this permission are able to run egosh debug and soamlog commands to change SOAM daemon debug levels across the cluster.
retrieve system log	Users with this permission are able to view the System Logs page via the PMC, and retrieve event and audit log files with access to all cluster data. Users can also run rfa commands.
resource plan control	Users with this permission are able to configure the resource plan. The extent of permission depends on the level of consumer tree specified with the role; if it is "/", the user can configure all resource plans, otherwise the user can only configure a plan for the specified consumer(s) and cannot insert or remove a time interval for time-based resource planning.

Permission	Description
deploy package control	Users with this permission are able to deploy service packages. The extent of permission depends on the level of consumer tree specified with the role; if it is "/", the user can deploy packages for all applications, otherwise the user can only deploy packages for the specified consumer(s). **Note:** As a best practice, a role with the deploy package control permission should be combined with a consumer user role to allow the user to log onto the PMC and deploy a service package.

Working with user roles

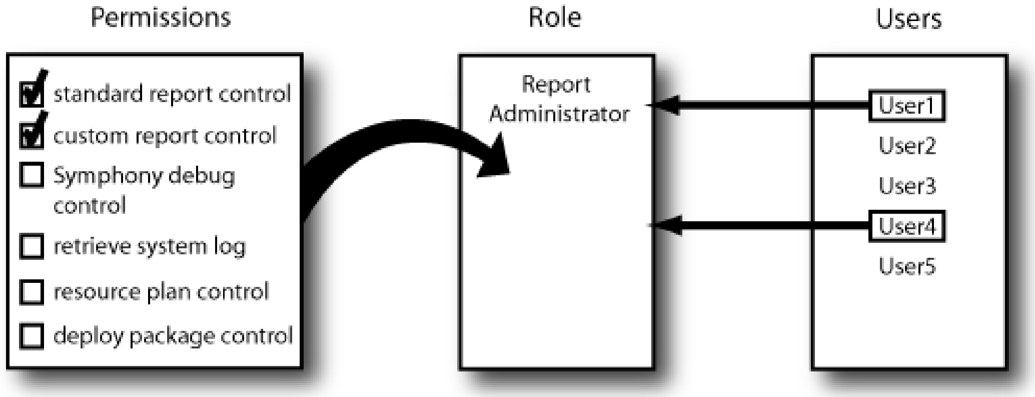
Roles can be assigned to any user and any user can have more than one role. If a user is assigned one of Symphony's preconfigured roles and a newly-created role, the effect is that the permissions of both roles are merged.

User role properties are configured through the Platform Management Console. You must be a cluster administrator to perform actions on user roles.

Perform the following steps when you want to implement a new user role:

1. Identify the role
2. Associate users with the role
3. Assign permissions to the role

The following example shows the concept of configuring one role for two users.



User1 and User4 are assigned the Report Administrator role.

The following properties of a user role can be modified using the PMC:

- description of the role
- users assigned to the role
- permissions

User roles can also be removed. Before removing a role, it is important to check that it is not assigned to any users, as this would cause them to lose all privileges associated with the role.

Create a user role

1. In the Platform Management Console, click Cluster > Configure User Roles.

   The User Role List page displays.
2. In the Global Actions dropdown list, select Create New Role.

   The Create New Role page displays.
3. In the User Role textbox, enter the name of the role.
4. In the Description textbox, enter a description of the role.
5. Select users from the Available User Accounts list to assign users to the role. Click Add.
6. Check the permission checkboxes to grant permissions to the role.
   a) If you want to grant all the available cluster-wide permissions, check the Cluster Permissions check box; otherwise, check the individual permission check boxes. Note that these permissions grant access to data and debug level controls for the whole cluster.
   b) If you want to grant consumer tree permissions, under Consumer Tree Permissions, click Add. In the tree list, select the consumer that you want to grant permissions to. Click Add.
   c) If you want to grant all the available consumer permissions, check the Consumer Permissions checkbox; otherwise, check the individual permission checkbox.
   d) If you want to grant permissions to other consumers: 1) click Add; 2) select the consumer from the consumer tree list; 3) choose the permission(s).
7. Click Create to save your changes.

Modify a user role

1. In the Platform Management Console, click Cluster > Configure User Roles.

   The User Role List page displays.
2. From the list of user roles, locate the role you want to modify and click it.

   The Role properties page displays.
3. Modify the user role property.
4. Click Apply to save your changes.

Remove a user role

1. In the Platform Management Console, click Cluster > Configure User Roles.

   The User Role List page displays.
2. From the list of user roles, locate the role you want to remove and select Actions > Delete role.

   Note that multiple roles cannot be removed at the same time.
3. Click OK.

Assign a role to a user

1. Follow the steps for modifying a user role.
2. In the Role properties page, under Specify users in this role, select the user from the list.
3. Click Add.

   The role is assigned to the user.

4. Click Apply.

Configure Windows password

If, when creating or modifying a consumer, you specify a Windows user account that has not already been configured, you have to set the password before you can continue.

Configure the password of every new Windows execution account that you introduce into the system. (The egoadmin account was configured during installation.)

Use the same command to update the system if the execution account password ever changes. You have to register the new password in EGO every time the execution account password changes in Windows.

Restriction:

The password must be 31 characters or less.

1. Log on to a host as egoadmin.
2. Log on to EGO as an EGO authentication user.

   egosh user logon -u *user_name* **-x** *password*

   For example, type

   ```
   egosh user logon -u Admin -x Admin
   ```
3. To configure the Windows password, run

   egosh ego execpasswd -u *user_name* -x *password*

   For example, if the account is mydomain\user2, type

   ```
   egosh ego execpasswd -u mydomain\user2 -x mypasswd
   ```

   Note:

   If you want to set the password for a Windows domain user from a Linux/UNIX host, enclose the Windows domain name in quotation marks:

   ```
   egosh ego execpasswd -u "mydomain\user2" -x mypasswd
   ```

Create a user account

You must be a cluster administrator to create a user account.

Create user accounts for each user you want to assign to a role for consumers. One user account can be assigned to multiple consumers with multiple roles. If the same user account is assigned more than one role to the same consumer, the highest role is assumed. Once you have created a user account, you can assign it a role when you create or modify a consumer.

1. Click Cluster > Configure User Accounts.

   A list of existing user accounts displays.
2. From Global Actions, select Create New User Account.

   The Create a New User Account page opens.
3. Fill in the fields.
   - The user name and password are mandatory.
   - The user name must be unique.
4. Click Create.

   Your new user account is displayed in the list.

You must assign this user account a role. You can assign it as a cluster administrator from Cluster > Summary > Cluster Properties or as a consumer user or consumer administrator in any consumer properties.

Delete a user account

You must be a cluster administrator to delete a user account.

User accounts can be deleted even if they are assigned a role in the cluster or for any consumer. A deleted user account is automatically removed from all consumers where it is assigned.

You cannot delete the default cluster administrator user account (`Admin`).

1. Click Cluster > Configure User Accounts.

   Any existing user accounts are listed.
2. Locate the user account you want to delete and click Actions > Delete User Account.
3. Confirm that you want to delete the user account.

Modify a user account

You must be a cluster administrator to modify a user account.

1. Click Cluster > Configure User Accounts.
2. Click the user account name you want to modify.

   The User Account Properties page opens.
3. Make any changes.

   You cannot change the user name.
4. Click Apply.

   Your modified user account displays in the list.

To change the roles that this user account has been assigned, you need to modify the consumer properties for each consumer where this user account has been assigned a role.

Add a cluster administrator

You must be a cluster administrator to create a user account or change cluster settings.

1. Navigate to Cluster > Summary.

   Cluster information displays in the Summary page.
2. Click Cluster Properties.

   Current cluster settings display in the Cluster Properties page.
3. Select the user account you want to have the cluster administrator role and click Add.
4. Click Apply.

Remove a cluster administrator

You must be a cluster administrator to delete a user account.

To remove a cluster administrator, you can delete their user account. Deleting a user account removes it from any role it may have been assigned everywhere in your consumer tree, including the cluster.

If you do not want to delete the user account, you can remove administrator privileges for that account in the cluster properties.

1. Click Cluster > Summary > Cluster Properties.

   In the second section, user accounts with cluster administrator privileges are on the right.
2. Use the Add and Remove buttons to modify the user accounts assigned as cluster administrators.
3. Click Apply.

Grant root privileges to a cluster administrator

Optional. A root user within a Linux/UNIX environment can choose to give root privileges within the cluster to the cluster administrator.

Check the following:

- That you are logged on as `root`.
- That `/etc/ego.sudoers` does not already exist. If the file does exist, use the -p option below.

By default, only root can start, stop, or restart the cluster.

Give root privileges to egoadmin so that egoadmin can start a local host in the cluster, or 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.

Do the following to give root privileges to egoadmin for one host. Run the command on each host in the cluster.

1. Run the `egosetsudoers.sh` command.

   Note:

   If you already have an `ego.sudoers` file from a previous cluster, run this command with the option -p.

   When you run `egosetsudoers.sh`, it does the following:

   It creates the `/etc/ego.sudoers` file. The file owner is `root` and the permissions are set to `600` because you ran this command as `root`. Only the root user can edit this file.

   It setuids the `egosh` command and change the owner of `egosh` to root.

   Whenever you see instructions to log on as root to start, stop, or restart a host in the cluster, you may log on as egoadmin instead.

Configure a secure Platform Management Console (https)

The `$JAVA_HOME` environment variable must be set correctly on the host where the WEBGUI service is running.

1. Stop the WEBGUI service:

   egosh service stop WEBGUI
2. Generate a self-signed certificate for internal use:
   1. Change directory to `$JAVA_HOME/bin`
   2. Type **./keytool -genkey -alias tomcat -keyalg RSA**

      A prompt for the keystore password displays.
   3. Use Tomcat's default password: **changeit**.
   4. When prompted for last and first name, enter the FQDN of the Platform Management Console host. For example:

      `sympmcserver.platform.com`
   5. Answer all other questions.
3. Enable the SSL.
   1. Find the SSL connector element in *`$EGO_CONFDIR/`*`../../gui/conf/server.xml`.

      The SSL connector element is located below this comment: `<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->`
   2. Uncomment the SSL connector element by removing the comment tags: <!--and -->.
   3. Change the connector port number to 8443 in `$EGO_CONFDIR/../../gui/conf/server.xml`.
4. Start the WEBGUI service:

   egosh service start WEBGUI

The Platform Management Console is now accessible through https and through port 8443. For example: https://*WEBGUI_server*:8443/Platform

Enabling Secure Shell

This feature allows the `egosh` command to use Secure Shell (SSH) to start the cluster instead of Remote Shell (RSH).

Configure SSH at the OS level

Refer to the FAQ in the Knowledge Center for configuration details.

Enable SSH

Grant root privileges to a cluster administrator. Enable SSH on the host from which you want to run egosh commands.

To enable SSH, perform the following configuration:

1. Define or edit the EGO_RSH parameter in *`$EGO_CONFDIR/`*`ego.conf` on the host from which you want to run the `egosh` command:

   EGO_RSH=ssh | “ssh $params”

   For example:

   EGO_RSH="ssh -o ’PasswordAuthentication no’ -o ’StrictHostKeyChecking no’"

   If you want to revert to RSH usage, remove the new line in `ego.conf` or update it, as follows:

   EGO_RSH=rsh

   Note:

   SSH must be configured correctly on all hosts. If the `egosh` command fails due to improper SSH configuration, the command will automatically be retried using RSH.

   Note:

   The user account of the user who starts the cluster must be able to run the ssh commands across all hosts.

Integrating Symphony with Microsoft Active Directory

Scope

The EGO Active Directory (AD) plug-in is available for the following platforms:

Management hosts	Compute hosts
Windows 2000, Windows 2003, and Windows 2008	All platforms supported by Symphony

Configure EGO and plug-in

1. Edit `adauth.conf` in $EGO_CONFDIR to set the values of mandatory and optional parameters. Set the value of mandatory parameter AD_ADMIN as the AD account that should be mapped to the EGO 'Admin' user. Here is an example of the `adauth.conf` file:

   ```
   # EGO AD plug-in configuration file
   # Mandatory parameters
   # AD_ADMIN=<ad-account-name>
   #          AD account that should be mapped to EGO 'Admin' user.
   #          Use the format <domain-name>\<username>
   # Optional parameters
   # AD_WINDOMAIN=<windows-domain1>,<windows-domain2> ...
   #          NetBIOS names of the Windows domains whose domain
   #          controllers are queried to obtain user list. If not
   #          specified, only the primary domain controller of EGO
   #          master host is queried. This parameter is not
   #          applicable if EGO master is a Linux host.
   # AD_CACHEEXPIRYTIME=<time-in-days>
   #          VEMKD caches the user list obtained from AD. This
   #          parameter specifies the expiry time of the user list
   #          cache. If not specified, default value of 1 day is
   #          used.
   AD_ADMIN=egoadmin
   #AD_WINDOMAIN=
   #AD_CACHEEXPIRYTIME=1
   ```
2. Change the password of the 'Admin' user in EGO using the utility 'egostashpass-AD'. The password should be the same as the password of the AD_ADMIN account specified in `adauth.conf` (in step 1.). EGO needs to store the password of this account as it uses this password to generate a credential for the EGO Service Controller and other EGO services.
3. Edit `ego.conf` on management hosts and modify the value of parameters EGO_SEC_PLUGIN and EGO_SEC_CONF as follows:

   - EGO_SEC_PLUGIN=sec_ego_ext_ad
   - EGO_SEC_CONF="C:\EGO\kernel\conf,0,INFO,C:\EGO\kernel\log"

     The format of EGO_SEC_CONF is <plugin-configuration-directory, created-ttl, plugin-log-level, plugin-log-directory>. All the server side messages will be logged to ego_ext_plugin_server.log in the plugin-log-directory.
4. Edit `ego.conf` on compute hosts and modify the value of parameters EGO_SEC_PLUGIN as follows:

   EGO_SEC_PLUGIN=sec_ego_ext_co

   EGO_SEC_CONF parameter is optional on compute and client hosts. Specify this parameter if log messages are required from the client-side plug-in. The format is the same as specified in step 3. The client-side messages will be logged to `ego_ext_plugin_client.log` in the plugin-log-directory.
5. Start the EGO cluster.

6. Optionally, add 'AD_ADMIN' user (see step 1) as Cluster Administrator after logging in as 'Admin'; otherwise, the ‘AD_ADMIN’ user will not have a role in the cluster.

Usage

Users can log into EGO using the same <domain>\<username> and password they use to log into a Windows machine in your organization. The EGO cluster administrator can log into EGO by using the username 'Admin' and the password of the mapping account specified in adauth.conf.

Note:

If you are logging into EGO on Unix, you must use two backslashes between the domain and user name, i.e., <domain>\\<username>.

Note:

For enhanced security, you can access the Platform Management Console using https. Refer to *Configure a secure Platform Management Console (https)* on page 302 for details.

Disable the AD plug-in

1. Unassign roles for all AD accounts in the cluster.
2. Edit `ego.conf` on management hosts and compute hosts by modifying the value of parameters EGO_SEC_PLUGIN and EGO_SEC_CONF as follows:
   - EGO_SEC_PLUGIN=sec_ego_ext_default
   - EGO_SEC_CONF=C:\EGO\kernel\conf (Windows)

     EGO_SEC_CONF=/opt/ego/kernel/conf (Linux/UNIX)
3. Restart EGO on the master hosts.

Note:

When logging on with Admin, use the new password. The new password is the same as AD_ADMIN after you run the `egostashpass-AD` command. You can change this password through the PMC or CLI.

Limitations

1. Operating Systems of all management hosts must use the same authentication mechanism.
2. Operating Systems of all management hosts must be Windows.
3. Before changing the security plug-in from non-AD plug-in to AD plug-in, roles for all users, with the exception of the built-in Admin user, should be unassigned in order to remove these users from the cluster. Although there is no impact to the cluster if roles remain assigned to these users, they will no longer be able to log on to the cluster.
4. The AD Plug-in does not support the Symphony GUI/CLI ability to add/delete/modify users.

CHAPTER

14

Using SSL Security with Symphony

About Symphony communication using SSL

The Secure Socket Layer (SSL) is a protocol that uses encryption and authentication techniques to secure connections between clients and servers.

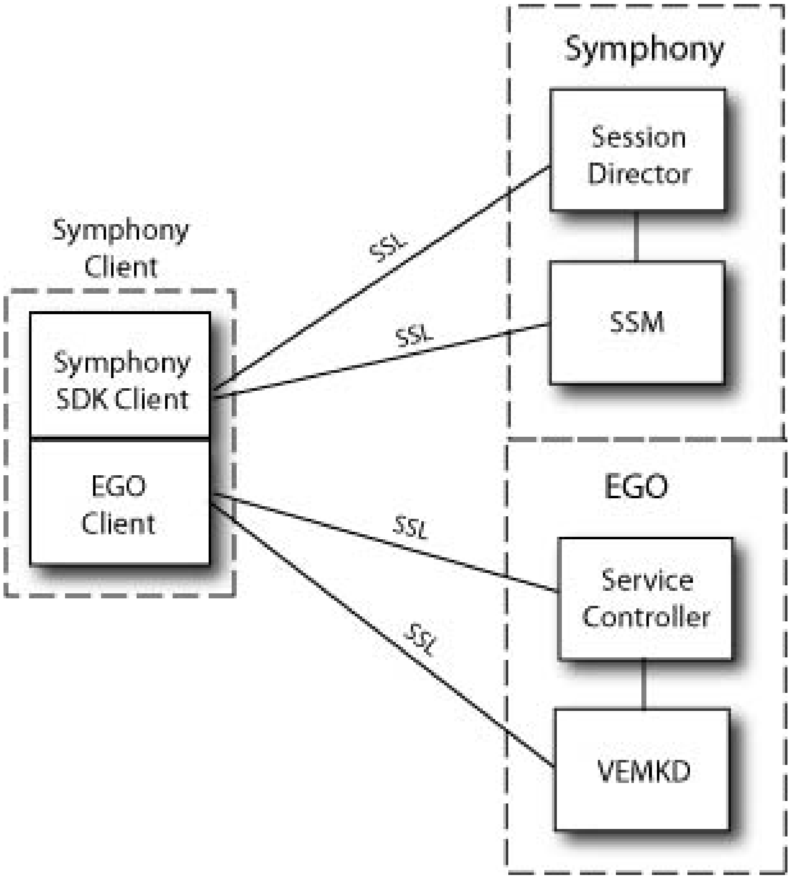
By encrypting data transfers between Symphony clients and servers, SSL virtually guarantees that if the data is intercepted enroute, it will be unintelligible without the relevant information required to decrypt it.

SSL also enables server authentication by using a certificate to validate its identity. This is especially important for preventing a 'man-in-the-middle' attack where someone actually pretends to be your server in order to gain access to sensitive material. SSL certificates ensure that the server you are connecting to is the server you intended to connect to.

About Symphony clients

Symphony uses EGO as its underlying resource allocation engine. A Symphony client not only connects to Symphony's Session Director (SD) and Session Manager (SSM) servers but to EGO's kernel daemon (VEMKD) and Service Controller, as well. This is because a Symphony client indirectly uses the EGO API to communicate with EGO. More specifically, a Symphony client is linked with the Symphony SDK library that uses the Symphony API (e.g. sessions). The Symphony API internally uses the EGO API to communicate with EGO, so the Symphony SDK client internally is also an EGO client.

To avoid ambiguity during the SSL configuration process, we will use the terms Symphony SDK client and EGO client in lieu of the generic Symphony client. Connections to both clients must be configured to ensure security for all communications between the clients and their respective servers.



How SSL works with Symphony

Client - server interactions

Since the connections between Symphony and its clients may be carrying sensitive information that is vulnerable to a security breach, there is a provision to safeguard the information by configuring the individual connections with SSL. Each connection can be configured independently and each server can implement a different authentication certificate. Note that Symphony DE does not support SSL configuration.

To better understand the implementation of SSL at Symphony startup, let's look at the sequence of events.

1. EGO starts the Service Controller.
2. The Service Controller starts the Session Director server as a service with its environment configured from its service definition file `sd.xml`. This file contains the SSL configuration for connections to the Symphony SDK client.
3. Upon startup, the Session Director listens on port *SD_SDK_PORT* for incoming connections from clients. The Session Director also registers with EGO as a client and uses its connection URL, <*SD-hostname*>:*SD_SDK_PORT*, as the client description. Note that *SD_SDK_PORT* is an environment variable defined in the `sd.xml` file.
4. The Symphony client opens a connection to EGO, which initiates SSL handshaking including server authentication, if so configured. EGO opens port *EGO_KD_TS_PORT*, which only accepts SSL connections. When transport security is set to SSL, client calls to the API use the *EGO_KD_TS_PORT*.
5. The Symphony client retrieves the connection URL to Session Director by passing the Session Director client name in an API call to EGO and retrieving the client description (connection URL).
6. The Symphony client connects to the Session Director.
7. The Session Director verifies the authentication information (e.g. name/password) and then starts an SSM server dedicated to the application. (The Session Director maintains an internal list with Application Profiles and starts the corresponding SSM as a service with its environment configured from `sd.xml`).
8. SSM finds the first available port and starts listening for clients. The SSM returns the URL as host:port to Session Director. Subsequent calls to the Session Director from new clients requesting an application that an SSM has already started will return the URL of the running SSM.
9. The Session Director sends the SSM's URL and a user credential to the client. At this point, the client is able to use the connection to send tasks.

SSL parameters

SOAM parameters

To configure SSL for individual connections between the Symphony client and Session Director or session manager, it is necessary to edit the `sd.xml` file. This section details the configurable parameters of the `sd.xml` file.

Session Director parameters

- *SD_SDK_TRANSPORT*: protocol driver. The driver value for SSL is "TCPIPv4SSL". If this parameter is not defined, the protocol driver is TCP/IPv.4 by default.
- *SD_SDK_TRANSPORT_ARG*: arguments for initializing the communication library (commLib). Arguments consist of security keys and certificates. The format for the arguments is the same as the one used in EGO_DEFAULT_TS_PARAMS and EGO_KD_TS_PARAMS. Alternatively, a variable, such as $EGO_DEFAULT_TS_PARAMS can be substituted in place of the arguments.

Session manager parameters

- *SSM_SDK_TRANSPORT*: protocol driver. The driver value for SSL is "TCPIPv4SSL".
- *SSM_SDK_TRANSPORT_ARG*: arguments for initializing the communication library (commLib). Arguments consist of security keys and certificates. The format for the arguments is the same as the one used in EGO_DEFAULT_TS_PARAMS and EGO_KD_TS_PARAMS. Alternatively, a variable, such as $EGO_DEFAULT_TS_PARAMS can be substituted in place of the arguments.

Client parameters

- *SDK_TRANSPORT*: protocol driver. The driver value for SSL is "TCPIPv4SSL".
- *SDK_TRANSPORT_ARG*: arguments for initializing the communication library (commLib). Arguments consist of security keys and certificates. The format for the arguments is the same as the one used in EGO_CLIENT_TS_PARAMS. Alternatively, the $EGO_CLIENT_TS_PARAMS variable can be substituted in place of the arguments.

EGO parameters

To configure SSL for individual connections between the EGO client and EGO, it is necessary to edit the `ego.conf` file on the management and client hosts as well as the `egosc_conf.xml` file on the EGO Service Controller host.

ego.conf parameters

This section details the configurable parameters of the `ego.conf` file. The SSL parameters in the `ego.conf` file must be configured for the VEMKD daemon and the client, whichever is applicable.

- EGO_TRANSPORT_SECURITY: (daemon and client) turns the transport security feature on or off.
- EGO_DEFAULT_TS_PARAMS: (daemon only) this is a general parameter consisting of subparameters defined for SSL that apply to every daemon and container in the cluster. If parameters are not defined, SSL will use anonymous DH as the cipher. The user can define daemon-specific parameters that override these default parameters. Refer to ego.conf subparameters for a list of applicable subparameters.
- EGO_KD_TS_PORT: (daemon and client) the SSL port number of VEMKD.

- EGO_KD_TS_PARAMS: (daemon only) the SSL parameters specific to VEMKD. Refer to ego.conf subparameters for a list of applicable subparameters.
- EGO_CLIENT_TS_PARAMS: (client only) the SSL parameters specific to the client. Refer to ego.conf subparameters for a list of applicable subparameters.

ego.conf subparameters

EGO_DEFAULT_TS_PARAMS, EGO_KD_TS_PARAMS, and EGO_CLIENT_TS_PARAMS contain the following configurable subparameters:

- CERTIFICATE: (daemon only) the location of the certificate file. Certificate files with the PEM file format are supported. For information about generating certificates using `openssl`, refer to http://www.openssl.org/docs/apps/openssl.html.
- CIPHER: (daemon and client) the cipher list used by SSL. The client and server will negotiate the cipher list and select the first shared one. The default list is ADH-DES-CBC3-SHA.
- CAFILE: (client only) the location of the Certification Authority (CA) certificate. The client reads this file and trusts the CA within the file. This parameter is used in cases where there is only one certificate file.
- CAPATH: (client only) the directory where the CA certificates are stored. This parameter is used when there are multiple CA files. It is a path that points to the directory where the files are stored.
- PRIVATE_KEY: (daemon only) the location of the private key file.
- SERVER_AUTH: (client only) defines whether client should authenticate the server and how to authenticate.

  Syntax:

  SERVER_AUTH=NONE|HOST|{string}name{string}…

  NONE: no server authentication is required. This is the default value.

  HOST: per host certificate, check the connected host with the subject CN (common name) in the certificate.

  {string}name{string}…:

  This format enables certificate verification on a per cluster, daemon or application basis.

  "name" can be either a name of a daemon (such as VEMKD) or an application (such as SOATesting). The string is the subject CN in the certificate. The first {string} is the default value for daemons/applications whose names are not defined here.

  Note:

  All Symphony daemons have reserved names. An application cannot have the same name as a Symphony daemon. For example, you cannot define a Symphony application with the name "vemkd".

  For Example:

  1. SERVER_AUTH={Platform EGO}: only default is provided. All daemons share the same certificate of "Platform EGO".

2. SERVER_AUTH=vemkd{Platform vemkd}egosc{Platform Service Controller}: value is provided for each daemon. Client will check VEMKD certificate with "Platform vemkd", and EGOSC certificate with "Platform Service Controller".
3. SERVER_AUTH={Platform EGO}SOATesting{SOA Testing}: both default and name-value pair are provided. Symphony client of SOATesting will check SSM certificate with "SOA Testing". All other clients check daemon certificate against "Platform EGO".

egosc_conf.xml parameters

The `egosc_conf.xml` file contains one configurable SSL parameter.

- ESC_TS_PARAMS: the SSL parameters for the EGO Service Controller. ESC_TS_PARAMS uses the same subparameters as EGO_KD_TS_PARAMS but they are applicable only to the Service Controller.

Configure security settings

Symphony's SSL functionality can be configured to suit specific security requirements.

In this section, we implement typical security settings, which enable server authentication and use a common SSL configuration for all servers and clients.

1. Open the `ego.conf` file on the management host using a text editor. The location of the file is defined in the *EGO_CONFDIR* environment variable.
2. Set the EGO_TRANSPORT_SECURITY parameter to SSL.
3. Set EGO_DEFAULT_TS_PARAMS.

   For example:

   - (Linux/UNIX)

     EGO_DEFAULT_TS_PARAMS="SSL[CERTIFICATE=/etc/symcert.pem,CIPHER=EDH-RSA-DES-CBC3-SHA,PRIVATE_KEY=/etc/symkey.pem]"
   - (Windows)

     EGO_DEFAULT_TS_PARAMS="SSL[CERTIFICATE=C:\xxc\newcert.pem,CIPHER=EDH-RSA-DES-CBC3-SHA,PRIVATE_KEY=C:\xxc\newkey.pem]"

   Note:

   For typical security requirements, do not define EGO_KD_TS_PARAMS and ESC_TS_PARAMS. In this case, VEMKD and the Service Controller will use the SSL parameters defined in EGO_DEFAULT_TS_PARAMS.
4. Assign an SSL port number to the EGO_KD_TS_PORT parameter.
5. Open the `ego.conf` file on the client host using a text editor.
6. For EGO_CLIENT_TS_PARAMS, enable server authentication.

   For example:

   - (Linux/UNIX)

     EGO_CLIENT_TS_PARAMS="SSL[CAFILE=/home/.../cacert.pem, CIPHER=EDH-RSA-DES-CBC3-SHA,SERVER_AUTH={myCN}"

   - (Windows)

     EGO_CLIENT_TS_PARAMS="SSL[CIPHER=EDH-RSA-DES-CBC3- SHA,CAFILE=C:\xxc\demoCA\cacert.pem,SERVER_AUTH={myCN}]"
7. Open the `sd.xml` file on the management host using an XML editor.
8. Set the SD_SDK_TRANSPORT parameter to **TCPIPv4SSL** (SSL driver on TCP/IPv.4).
9. Set the SD_SDK_TRANSPORT_ARG parameter to **$EGO_DEFAULT_TS_PARAMS**.
10. Set SSM_SDK_TRANSPORT parameter to **TCPIPv4SSL** (SSL driver on TCP/IPv.4).
11. Set SSM_SDK_TRANSPORT_ARG parameter to **$EGO_DEFAULT_TS_PARAMS**.
12. Set the SDK_TRANSPORT parameter to **TCPIPv4SSL** (SSL driver on TCP/IPv.4).
13. Set SDK_TRANSPORT_ARG to **$EGO_CLIENT_TS_PARAMS**.

Sample configuration

This section provides a sample configuration using typical security settings where all daemons share one certificate.

```
# in ego.conf on master and management host(s), add:
EGO_TRANSPORT_SECURITY=SSL
EGO_KD_TS_PORT=32779 (user has to define port number here)
EGO_DEFAULT_TS_PARAMS="SSL[CERTIFICATE=/etc/.../newcert.pem,CIPHER=EDH-RSA-DES-CBC3-
SHA,PRIVATE_KEY=/etc/.../newkey.pem]"
EGO_CLIENT_TS_PARAMS="SSL[CIPHER=EDH-RSA-DES-CBC3-SHA,CAFILE=/etc/.../
demoCA/cacert.pem,SERVER_AUTH={myCN}]"
# in ego.conf on client host(s), add:
EGO_TRANSPORT_SECURITY=SSL
EGO_KD_TS_PORT=32779 (user has to define port number here)
EGO_CLIENT_TS_PARAMS="SSL[CIPHER=EDH-RSA-DES-CBC3-SHA,CAFILE=/etc/.../
demoCA/cacert.pem,SERVER_AUTH={myCN}]"
# in the <ego:ContainerSpecification> section in sd.xml on master/master-failover
# host(s), add:
<ego:EnvironmentVariable name="SD_SDK_TRANSPORT">TCPIPv4SSL</ego:EnvironmentVariable>
<ego:EnvironmentVariable name="SD_SDK_TRANSPORT_ARG">$EGO_DEFAULT_TS_PARAMS</
ego:EnvironmentVariable>
<ego:EnvironmentVariable name="SSM_SDK_TRANSPORT">TCPIPv4SSL</ego:EnvironmentVariable>
<ego:EnvironmentVariable name="SSM_SDK_TRANSPORT_ARG">$EGO_DEFAULT_TS_PARAMS</
ego:EnvironmentVariable>
<ego:EnvironmentVariable name="SDK_TRANSPORT">TCPIPv4SSL</ego:EnvironmentVariable>
<ego:EnvironmentVariable name="SDK_TRANSPORT_ARG">$EGO_CLIENT_TS_PARAMS</
ego:EnvironmentVariable>
```

CHAPTER

15

Configuring a Mixed OS Cluster

The cluster must have a master host and the management hosts all running the same operating system. If you install compute hosts with operating systems that differ from the master host and the management hosts, you need to install a mixed cluster. A mixed cluster is useful if you have cross-platform applications that can run on either Linux/UNIX or Windows.

If you are installing a cluster with a Linux/UNIX master host, follow the installation instructions to install Linux/UNIX management hosts and compute hosts. You will need to follow this chapter to install Windows or Solaris compute hosts.

If you are installing a cluster with a Windows master host, follow the installation instructions to install Windows management hosts and compute hosts. You will need to follow this chapter to install Linux/UNIX compute hosts.

Workload execution account

For a consumer to execute work on Linux/UNIX and Windows hosts, you need one Linux/UNIX execution user account and one Windows execution user account with the same user name. For example, if the actual Windows account is `DOMAIN\test06` or `.\test06`, the Linux/UNIX account is `test06`.

You always want the Windows account name as the execution user in the consumer properties. If the execution host is Linux/UNIX, the domain name is automatically stripped (for example, `DOMAIN\test06` is interpreted as `test06` on Linux/UNIX).

Even if you do not run cross-platform applications, you will need to modify some built-in consumers when creating to a mixed-OS cluster.

With advanced workload execution mode, for every new consumer that you create in your cluster, input the Windows account name when you configure the execution user in the consumer properties. For other cases, you need to edit `ConsumerTrees.xml` manually to change this configuration.

Setting workload execution account manually

For existing consumers configured with the Linux/UNIX user name only, take these steps to configure the workload execution account manually.

These steps describe how to add the Windows domain name to the configuration. Do not change the execution user to a different user account, it is likely to cause problems.

1. Edit $EGO_CONFDIR\`ConsumerTrees.xml`
2. For each consumer that needs to run cross-platform workload, edit the `<ExecutionUser>` section. Add the Windows domain name to the existing UNIX user name (for example, change `test06` to `DOMAIN\test06`).
3. Save and close the file.
4. Restart EGO on the master host.

What you need to know

To expand the cluster, you can:

- add Linux/UNIX compute hosts to your Windows cluster
- add Windows compute hosts to your Linux/UNIX cluster
- add UNIX/Linux and Windows compute hosts to your Solaris cluster

You may also mix multiple Linux/UNIX host types in a cluster. These kinds of mixed clusters are for advanced users.

When you plan a mixed cluster, you must be aware of the following points.

Cluster administrator account

Linux/UNIX

To support Linux/UNIX hosts, the cluster requires a user account as cluster administrator (the default is `egoadmin`). The cluster administrator account must exist on every Linux/UNIX host and have the same name as the Windows cluster administrator account. For example, if the actual Windows account name is `.\newadmin` or `DOMAIN\newadmin`, the Linux/UNIX account name is `newadmin`.

The cluster administrator is the only non-root account that has permission to manage Linux/UNIX hosts in the cluster. The account owns most Symphony files on Linux/UNIX hosts.

You must create the cluster administrator account, or have a valid user account be the cluster administrator, before you start the installation.

By default, only the `root` account can start, stop, or restart hosts. You can assign root privileges to the cluster administrator account.

Windows cluster

To support Windows hosts, the cluster requires a Windows OS user account as the cluster administrator (the default is `egoadmin`). The Windows cluster administrator account must exist on every Windows host and have the same name as the Linux/UNIX cluster adminitrator account. For example, if the actual Linux/UNIX account name is `newadmin`, the Windows account can be named `.\newadmin` or `DOMAIN\newadmin`, but cannot be named `.\user4` or `DOMAIN\user4`.

The Windows cluster administrator is the only account that can start up, restart, or shut down all Windows hosts in the cluster. The Local Administrator of the host can start and stop Symphony services on the host as well.

Both the cluster administrator and the local administrator have full control over all Symphony files.

You must create the cluster administrator account, or have a valid user account be the cluster adminstrator, before you start the installation. The account requires certain privileges and permissions as described in the installation guide.

Installation directory

The installation directory is the directory where the Symphony binaries are installed on a host.

The default installation directory on Linux/UNIX hosts is `/opt/ego`. This does not need to have the same name as the Windows installation directory, but it must be the same directory in all Linux/UNIX hosts.

The default installation directory on Windows hosts is `C:\EGO`. This does not need to have the same name as the Linux/UNIX installation directories, but it must be the same directory in all Windows hosts.

To use a different directory, you must customize the installation.

Linux and Windows installers create the installation directory if it does not already exist. If it does exist, make sure the directory is empty.

On UNIX hosts, you need to create the installation directory prior to installation.

Execution accounts for consumers installed with Symphony

Consumers installed with Symphony are configured to use the cluster administrator account as the execution user. Normally, you should not edit any of these consumers.

However, for certain consumers to execute work on both Windows and Linux/UNIX hosts, you need to make sure the execution user in the consumer properties uses the Windows account name. If the execution host is Linux or UNIX, the domain name is automatically stripped (for example, `DOMAIN\egoadmin` is interpreted as `egoadmin` on Linux and UNIX).

If the master host runs Windows, the consumers are already configured with the Windows user name, do not make any changes. If the master host runs Linux/UNIX, you must edit `ConsumerTrees.xml` manually, and for each of these consumers add the Windows domain name to the existing Linux/UNIX user name (for example, change `egoadmin` to `DOMAIN\egoadmin`).

You must also register the cluster administrator password.

The consumers you need to modify are:

- `EGOClusterServices` under `ClusterServices`
- `SymphonyClusterServices` under `ClusterServices`
- `Symping51`
- `SymExec51`

Also, if you will use any of the sample applications, modify the execution account of the consumer before you deploy the related sample application.

Consistent configuration on all hosts

When installing a mixed cluster, make sure that you use the same configuration for the following settings in all hosts on each operating system.

- Use the same base port as all other hosts in the cluster.

  To check the base port, see EGO_LIM_PORT in the `ego.conf` file on the master host.
- Use the same cluster name as all other hosts in the cluster.

  The cluster name is displayed in the Platform Management Console and is part of the file name `ego.cluster.`*`cluster_name`* on the master host.
- In addition, when you install on the Solaris host, you need to consider the following:
  - Use the same Symphony daemon communication ports as the other hosts.

    To check the Symphony daemon communication ports, see EGO_KD_PORT and EGO_PEM_PORT in the `ego.conf` file on the master host.
  - Use the same list of master candidate hosts as the other hosts.

    To check the list of master candidate hosts, see EGO_MASTER_LIST in the `ego.conf` file in the EGO configuration directory:

    For a Linux/UNIX master host, check `$EGO_CONFDIR/ego.conf`.

For a Windows master host, check `%EGO_CONFDIR%\ego.conf`.

Install a Windows compute host in a mixed cluster

Add Windows compute hosts to a Linux/UNIX cluster.

Configure the cluster administrators

Log on as `egoadmin`.

For proper administration of a mixed cluster, configure the cluster to recognize both Linux/UNIX and Windows cluster administrator accounts.

1. Open the cluster file for editing.

   `$EGO_CONFDIR/ego.cluster.`*`cluster_name`*
2. Find the ClusterAdmins section.

   ```
   Begin ClusterAdmins
   ...
   End ClusterAdmins
   ```
3. Add the Windows cluster administrator account to the Administrators parameter (which already contains the Linux/UNIX cluster administrator account).

   For example:

   ```
   Begin ClusterAdmins
   ...
   Administrators=egoadmin domain\egoadmin
   ...
   End ClusterAdmins
   ```
4. Save the file.
5. Restart the master host.

   ```
   egosh ego restart HostM
   ```

Installing Symphony on a Windows host

The Windows hosts in your cluster must be one of the following supported versions:

- Windows 2003
- Windows 2000
- Windows XP

It is not necessary that all Windows hosts run the same version of Windows.

Install the software on each Windows compute host by following the installation guide.

When you install on Windows hosts, joining the cluster and starting the host is an automatic part of the installation process. There is no need to configure a host after installation is complete.

Run the test application

A test application that sends Symphony workload to your cluster is installed with your system. This application is already pre-deployed to all compute hosts and is included in your path.

Note:

Before running the test application, you must configure the required consumer by adding the Windows domain name to the existing Linux/

UNIXuser name (for example, change egoadmin to DOMAIN\egoadmin). Refer to *Workload execution account* on page 318 for more information.

Run the test application to ensure your system is working properly.

1. Run the test application

   For example

   symping

   If the system is working properly, the output shows the task summary and detail reports.

Install a Linux compute host in a mixed cluster

Add Linux compute hosts to a Windows cluster.

Configure the cluster administrators

Log on as `egoadmin`.

For proper administration of a mixed cluster, configure the cluster to recognize both Linux/UNIX and Windows cluster administrator accounts.

1. Open the cluster file for editing.

   `$EGO_CONFDIR/ego.cluster.`*`cluster_name`*
2. Find the ClusterAdmins section.

   ```
   Begin ClusterAdmins
   ...
   End ClusterAdmins
   ```
3. Add the Linux/UNIX cluster administrator account to the Administrators parameter (which already contains the Windows cluster administrator account).

   For example:

   ```
   Begin ClusterAdmins
   ...
   Administrators=domain\egoadmin egoadmin
   ...
   End ClusterAdmins
   ```
4. Save the file.
5. Restart the master host.

   ```
   egosh ego restart HostM
   ```

Installing Symphony on a Linux host

Install the software on each Linux compute host by following the installation guide.

When you install on Linux hosts, joining the cluster and starting the host is a manual process. The installation process is not complete until you configure a host.

Run the test application

Run the test application to ensure your system is working properly.

1. Run the test application

   To force the application to run on all hosts, use `-m` to specify the maximum number of slots available to the `SymTesting` consumer.

   For example, if there are 25 single-slot compute hosts:

   symping -m 25 -r 60000

   If you log onto the Platform Management Console at this stage, you can see the work running on your hosts, as long as you temporarily reset the filter settings in the task view page.
2. Check hosts.

   Check the output to see what hosts were used. The test application can run on both Windows and Linux compute hosts.

Install a Solaris compute host

Add Solaris compute hosts to a Linux or Windows cluster.

Installing Symphony on a Solaris host

Install the software on each Solaris compute host.

Different versions of Solaris have different installers. To install a .bin package, ignore the steps in this document and follow the usual procedure described in the Symphony installation documentation.

To install a .tar package, follow the steps in this document. When you use the .tar package, joining the cluster and starting the host is a manual process. The installation process is not complete until you configure a host.

Deploy the EGO software on the compute host

Check the following:

- That you are logged on as the cluster administrator.

  For example, the default cluster administrator for Linux and Windows hosts is `egoadmin`.

  If you cannot be `egoadmin`, you must set `CLUSTERADMIN` as a valid OS account.
- That you created the installation directory and copied the EGO installation package into this directory.

  For example, you created `/opt/ego`.
- That the required connection ports are not in use. You must use the same ports as the hosts in your cluster.

  The default base connection port is 7869. Symphony uses four consecutive ports from this base port (7869-7872)

Complete the following steps to deploy the software.

1. Extract the EGO installation package.
2. Modify the EGO environment configuration files.
3. Set the command-line environment.
4. Enable automatic startup.
5. Grant root privileges to a cluster administrator (Optional).

Extract the EGO installation package

Run the EGO installation package on your compute host.

1. Navigate to the installation directory that you created.

   For example:

   cd /opt/ego
2. Extract the EGO Solaris installation package to the installation directory.

```
gzip -d -c egocomputehost-platform_type-sol***.tar.gz | tar xvf -
```

Modify the EGO environment configuration files

1. Navigate to the EGO installation directory.

   For example:

```
cd /opt/ego
```

2. Modify the top-level environment script file for your particular shell environment.
   a) Edit the `cshrc.platform` or `profile.platform` environment script file.
      - For `csh` or `tcsh`, edit the `cshrc.platform` file.
      - For `sh`, `ksh`, or `bash`, edit the `profile.platform` file.

   b) Replace all instances of `@EGO_TOP@` with the path to your installation directory.

      For example, replace "`@EGO_TOP@`" with "`/opt/ego`".
3. Navigate to the EGO configuration directory.

   For example:

   ```
   cd /opt/ego/kernel/conf
   ```
4. Modify the environment script file for your particular shell environment.
   a) Edit the `cshrc.ego` or `profile.ego` environment script file.
      - For `csh` or `tcsh`, edit the `cshrc.ego` file.
      - For `sh`, `ksh`, or `bash`, edit the `profile.ego` file.

   b) Replace all instances of `@EGO_TOP@` with the path to your installation directory.

      For example, replace "`@EGO_TOP@`" with "`/opt/ego`".
5. Modify the EGO configuration file (`ego.conf`).
   a) Edit the `ego.conf` file.
   b) Replace all instances of `@EGO_TOP@` with the path to your installation directory.

      For example, replace "`@EGO_TOP@`" with "`/opt/ego`".

   c) Define the master candidate list and EGO daemon port parameters using the same values as defined in the `ego.conf` file on the master host.

      You need to define the following parameters:
      - EGO_MASTER_LIST
      - EGO_LIM_PORT
      - EGO_KD_PORT
      - EGO_PEM_PORT

Set the command-line environment

On Solaris hosts, set the environment before you run any Symphony commands. You need to do this once for each session you open. Both `root` and the cluster administrator accounts use Symphony commands to configure and start the cluster.

These examples assume that your installation directory is `/opt/ego`.

- For `csh` or `tcsh`, use `cshrc.platform`:

  ```
  source /opt/ego/cshrc.platform
  ```
- For `sh`, `ksh`, or `bash`, use `profile.platform`:

  ```
  . /opt/ego/profile.platform
  ```

Enable automatic startup

Optional. By default, you must start EGO manually if a host restarts. For ease of administration, you should enable automatic startup. This feature starts EGO automatically when the host restarts.

1. Log onto the host as the cluster administrator.
2. Modify the EGO daemons script file.
    a) Edit the `ego_daemons.sh` file located in the `/etc` subdirectory of your EGO installation directory.

       For example, edit `/opt/ego/`*`version_number`*`/sparc-sol7-64/etc/ego_daemons.sh`.
    b) Replace all instances of `@EGO_TOP@` with the path to your installation directory.

       For example, replace "`@EGO_TOP@`" with "`/opt/ego`".
3. Log onto the host as `root`.
4. Run the `egosetrc.sh` command.

   Enabling automatic system startup creates an `ego` link under `/etc/rc.d/init.d`.

Grant root privileges to a cluster administrator

Optional. A root user within a Solaris environment can choose to give root privileges within the cluster to the cluster administrator.

By default, only `root` can start, stop, or restart the cluster.

Give root privileges to the cluster administrator so that it can start a local host in the cluster, or shut down or restart any hosts in the cluster from the local host. For the cluster administrator 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.

Do the following to give root privileges to the cluster administrator for one host. Run the command as `root` on each host in the cluster.

1. Run the `egosetsudoers.sh` command.

   Note:

   If the /etc/ego.sudoers file already exists, run `egosetsudoers.sh -p` instead.

   When you run `egosetsudoers.sh`, it does the following:

    1. It creates the `/etc/ego.sudoers` file. The file owner is `root` and the permissions are set to `600` because you ran this command as `root`. Only the `root` user can edit this file.
    2. It will setuid the `egosh` command and change the owner of `egosh` to root.

   Whenever you see instructions to log on as `root` to start, stop, or restart a host in the cluster, you may log on as the cluster administrator instead.

Users listed in `EGO_STARTUP_USERS` are now able to run the commands to start, stop, or restart a host in the cluster as `root`.

Deploy the Symphony software on the compute host

Check the following:

- That you are logged on as the cluster administrator.

  For example, `egoadmin`.
- That the installation directory exists and you copied the Symphony installation package into this directory.

For example, `/opt/ego`.

- That the required connection ports are not in use. You must use the same ports as the hosts in your cluster.

  The default base connection port is 7869. Symphony uses four consecutive ports from this base port (7869-7872)

Complete the following steps to deploy the software.

1. Extract the Symphony installation package.
2. Modify the Symphony environment configuration files.
3. Start the host.
4. Test that the host was added to the cluster.
5. Run the test application.

Extract the Symphony installation package

Run the Symphony installation package on your compute host.

1. Navigate to the installation directory.

   For example:

   cd /opt/ego
2. Extract the Symphony Solaris installation package to the installation directory.

```
gzip -d -c symcomputehost-platform_type-sol***.tar.gz | tar xvf -
```

Modify the Symphony environment configuration files

1. Navigate to the installation directory.

   For example:

   cd /opt/ego/
2. Modify the environment script file for your particular shell environment.
   a) Edit the `cshrc.platform` or `profile.platform` environment script file.
      - For `csh` or `tcsh`, edit the `cshrc.platform` file.
      - For `sh`, `ksh`, or `bash`, edit the `profile.platform` file.

   b) At the end of the file, add a command to set the environment using the Symphony scripts.

      The Symphony scripts are in the Symphony configuration directory.

      For example:

      - For `csh` or `tcsh`, add the following line:

        ```
        source /opt/ego/soam/conf/cshrc.soam
        ```
      - For `sh`, `ksh`, or `bash`, add the following line:

        ```
        . /opt/ego/soam/conf/profile.soam
        ```
3. Navigate to the Symphony configuration directory.

   For example:

   cd /opt/ego/soam/conf
4. Modify the environment script file for your particular shell environment.

a) Edit the `cshrc.soam` or `profile.soam` environment script file.

- For `csh` or `tcsh`, edit the `cshrc.soam` file.
- For `sh`, `ksh`, or `bash`, edit the `profile.soam` file.

b) Replace all instances of `$SOAM_HOME` with the path to the `soam` subdirectory of your installation directory.

For example, replace "`$SOAM_HOME`" with "`/opt/ego/soam`".

Set the command-line environment

On Solaris hosts, set the environment before you run any Symphony commands. You need to do this once for each session you open. Both `root` and the cluster administrator accounts use Symphony commands to configure and start the cluster.

These examples assume that your installation directory is `/opt/ego`.

- For `csh` or `tcsh`, use `cshrc.platform`:

```
source /opt/ego/cshrc.platform
```

- For `sh`, `ksh`, or `bash`, use `profile.platform`:

```
. /opt/ego/profile.platform
```

Start the host

Start Symphony on the host.

You must be logged on as `root` (or the cluster administrator with root privileges).

To start the local Platform Symphony host, perform the following steps:

1. Start Platform Symphony:

```
egosh ego start
```

You have now started Symphony on the host.

Test that the host was added to the cluster

You installed EGO on the host. You are logged on as the cluster administrator.

1. Run `egosh resource list` to see the resources in your cluster.

   Look for the host you added in the list of the resources.

   For example:

```
egosh resource list
NAME      status    mem    swp   tmp    ut  it   pg    r1m   r15s  r15m  ls
hostM     ok        176M   619M 4819M  1%   0   2.8   0.0   0.0   0.0   2
hostC     unavail   173M   635M 4800M  0%   0   0.9   0.0   1.3   0.0   1
```

If you can see the host name in the list of resources, that host was successfully added to the cluster.

This test detects hosts even if the host is not currently available. Some hosts make take a while to become available after they are added to the cluster.

Run the test application

A test application that sends Symphony workload to your cluster is installed with your system. This application is already pre-deployed to all compute hosts and is included in your path. The required consumer is already configured in your cluster.

Run the test application to ensure your system is working properly.

Note:

The application should be set to run in the global zone.

1. Run the test application

   For example

   ```
   symping
   ```

   If the system is working properly, the output shows the task summary and detail reports.

CHAPTER

16

Hosts with More Than One IP Address, Threads, Cores, or CPUs

Multi-homed hosts

Hosts that have more than one network interface usually have one Internet Protocol (IP) address for each interface. Such hosts are called multi-homed hosts.

EGO identifies hosts by their official host name, so it needs to match each of the network addresses of multi-homed hosts with a single host name. To do this, the host name information must be configured so that all of the Internet addresses for a host resolve to the same name.

Multiple network interfaces

Some system manufacturers recommend that each network interface, and therefore, each Internet address, be assigned a different host name. Each interface can then be directly accessed by name. This setup is often used to make sure NFS requests go to the nearest network interface on the file server, rather than going through a router to some other interface. Configuring this way can confuse EGO, because there is no way to determine that the two different names (or addresses) mean the same host.

All host naming systems can be configured so that host address lookups always return the same name, while still allowing access to network interfaces by different names. Each host has an official name and a number of aliases, which are other names for the same host. By configuring all interfaces with the same official name but different aliases, you can refer to each interface by a different alias name while still providing a single official name for the host.

IP connectivity

Some or all hosts have multiple network interfaces that connect to physically segmented networks. You may not want EGO to use the first IP address according to DNS to initiate a connection.

IP preference

A host has multiple network interfaces that connect to physically connected networks, but for routing or performance reasons, you might want to assign network interface preferences to different activities.

For example, communication between a client and management hosts could use one network interface, and communication between compute and management hosts could use another network interface. While it might be physically possible for a socket client to use the first IP address of a socket server according to DNS to initiate a connection, this might not be desirable.

Host name lookup

A common DNS server may return a different IP address in host name lookups depending on which subnet that host is on (different BIND/DNS views). For example, host named hostA might resolve to 192.168.0.1 on one subnet and 10.0.0.1 on another subnet on the same network.

Filtering a preferred IP address from multiple IP addresses

Use EGO_PREFERRED_IP_MASK in `ego.conf` to specify the preferred IP address for multiple network interfaces.

If more than one IP address matches the IP mask, the first matching IP address is used as the preferred IP address. If no addresses match the mask, the order of the address list is not changed.

Under some circumstances (when you have multiple aliases), you also need to specify the unique official name and list the aliases.

IP mask format

Specify the IP mask as conventional CIDR blocks, consisting of a four-part dotted-decimal address, followed by a slash, then a number from 0 to 32:

```
nnn.nnn.nnn.nnn/nn
```

The number following the slash is the prefix length, which is the number of shared initial bits, counting from the left-hand side of the address. An IP address matches the CIDR prefix if the initial *nn* bits of the address and the CIDR prefix are the same.

For example, if the IP mask is defined as 10.0.0.0/8, and the address list contains

192.168.0.1

10.1.1.1

192.168.1.1

The address list is reordered to reflect the IP preference:

10.1.1.1

192.168.0.1

192.168.1.1

Identifying aliases

The aliases and associated IP address must be listed in the `hosts` file under `EGO_CONFDIR`.

Note:

The configuration of the hosts file in the EGO_CONFDIR directory only affects the behavior of Symphony daemons and clients. It has no impact on OS commands or user applications.

Note:

The hosts file in the EGO_CONFDIR directory takes precedence over the OS hosts file.

Example

- You have two networks: NetA 172.0.0.* and NetB 10.0.0.*
- The client is on NetA but it cannot access NetB.
- The client host name is Client1 and it has one IP address: 172.0.0.200.
- The management hosts are linked to both networks.
- The master host (also running some other system services) has one official host name (MgmtHost1) but two aliases:
  - MgmtHost1a (172.0.0.1)
  - MgmtHost1b (10.0.0.1)
- SSM host has one official name (MgmtHost2) but two aliases:
  - MgmtHost2a (172.0.0.2)
  - MgmtHost2b (10.0.0.2)
- Compute hosts are linked to NetB network with one host name (ComputeHost*) and with one IP address (10.0.0.*).

- NetB is faster and preferred.

On the client, you would set the following:

1. In `$EGO_CONFDIR/ego.conf`, set **EGO_PREFERRED_IP_MASK="172.0.0.0/8"**
2. In `$EGO_CONFDIR/hosts`, add the following lines:

   172.0.0.1 MgmtHost1 MgmtHost1a

   172.0.0.2 MgmtHost2 MgmtHost2a

On the grid, you would set the following:

1. In `$EGO_CONFDIR/ego.conf`, set **EGO_PREFERRED_IP_MASK="10.0.0.0/8"**
2. In `$EGO_CONFDIR/hosts`, add the following lines:

   ```
   172.0.0.1 MgmtHost1 MgmtHost1a
   10.0.0.1 MgmtHost1 MgmtHost1b
   172.0.0.2 MgmtHost2 MgmtHost2a
   10.0.0.2 MgmtHost2 MgmtHost2b
   ```

Result:

The client can connect to 172.0.0.* hosts but cannot access 10.0.0.* hosts. (Client cannot access the grid.)

Configure IP preference for multi-homed hosts

For network routing or performance reasons, you might want to assign network interface preferences to different activities. For example, if you want to specify a preferred IP address for initiating connections from compute hosts to management hosts and from management hosts to compute hosts.

1. Edit `ego.conf`.
2. Set EGO_PREFERRED_IP_MASK="*ip_mask*".

   where *ip_mask* is used to reorder the list of IP addresses for a host name so that a preferred IP address as determined by the mask is put first in the list, and that address is used to initiate the connection to the host.

   For example:

   ```
   EGO_PREFERRED_IP_MASK="10.0.0.0/8"
   ```
3. Save `ego.conf`.
4. Shut down and restart your cluster. Refer to *Shut down Symphony cluster* on page 59 and *Start Symphony cluster* on page 58.

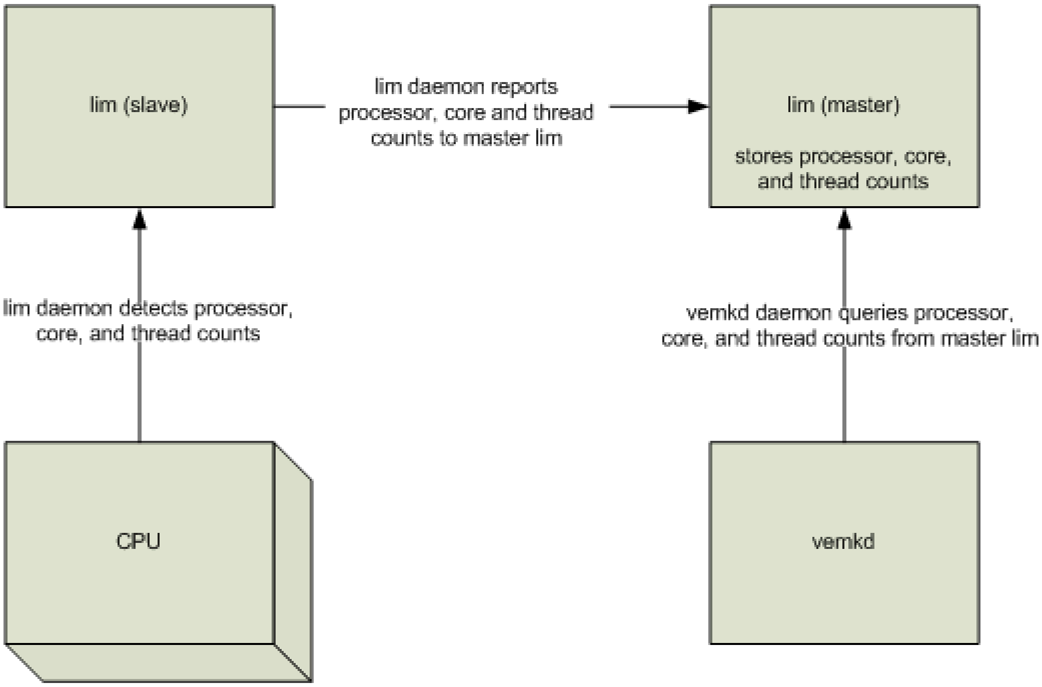
Understanding how the lim daemon detects cores, threads, and processors

Traditionally, the value of ncpus has been equal to the number of physical CPUs. However, most CPUs consist of multiple cores and threads, so the traditional 1:1 mapping is no longer useful. A more useful approach is to set ncpus to equal one of the following:

- The number of processors
- Cores: the number of cores (per processor) * the number of processors (this is the ncpus default setting)
- Threads: the number of threads (per core) * the number of cores (per processor) * the number of processors

An EGO cluster administrator globally defines how ncpus is computed using the EGO_DEFINE_NCPUS parameter (instead of EGO_ENABLE_DUALCORE) in `ego.conf` (shared directory).

The lim daemons detect and store the number of processors, cores, and threads for all supported architectures. The following diagram illustrates the flow of information between daemons, CPUs, and other components.


lim (slave)
lim daemon reports processor, core and thread counts to master lim
lim (master)
stores processor, core, and thread counts
lim daemon detects processor, core, and thread counts
vemkd daemon queries processor, core, and thread counts from master lim
CPU
vemkd

Although the ncpus computation is applied globally, it can be overridden on a per-host basis.

To correctly detect processors, cores, and threads, lim daemons/services assume that all physical processors on a single machine are of the same type.

In cases where CPU architectures and operating system combinations may not support accurate processor, core, thread detection, lim uses the defaults of 1 processor, 1 core per physical processor, and 1 thread per core. If lim detects that is it is running in a virtual environment (for example, VMware®), each detected processor is similarly reported (as a single-core, single-threaded, physical processor).

Lim detection uses processor- or OS-specific techniques (for example, Intel’s CPUID instruction, or Solaris’ kstat()/core_id). Note that if the operating system doesn't recognize a CPU or core (for example, if an older OS does not recognize a quad-core processor and instead detects it as dual-core), then the lim won't recognize it either; the lim only detects hardware that is recognized by the operating system.

Note:

RQL normalization never considers threads. Consider a hyper-thread enabled Pentium: Threads are not full-fledged CPUs, so considering them as CPUs would artificially lower the system load.

On a machine running AIX, detection of ncpus is different. Under AIX, the number of detected physical processors is always 1, whereas the number of detected cores is always the number of cores across all physical processors. Thread detection is the same as other operating systems (the number of threads per core).

Define ncpus—processors, cores, or threads

An EGO cluster administrator must define how ncpus is computed. Usually, the number of available EGO slots is equal to the value of ncpus; however, slots can be redefined at the resource group level. The ncpus definition is globally applied across the cluster.

1. Open `ego.conf`.
   - Linux/UNIX: `$EGO_CONFDIR/ego.conf`
   - Windows: `%EGO_CONFDIR%\ego.conf`
2. Define the parameter EGO_DEFINE_NCPUS=[procs | cores | threads].

   Set it to one of the following:

   - procs (where ncpus=procs)
   - cores (where ncpus=procs *cores)
   - threads (where ncpus=procs *cores * threads)

   By default, ncpus is set to **cores**.

   Note:

   In clusters with older lim daemons/services that do not recognize cores and threads, this parameter is ignored. In clusters where only the master lim recognizes cores and threads, the master lim assigns defaults values.
3. Save and close `ego.conf`.
4. Restart EGO.

Note:

As a best practice, set EGO_DEFINE_NCPUS instead of EGO_ENABLE_DUALCORE. The functionality of EGO_ENABLE_DUALCORE=y is preserved by setting EGO_DEFINE_NCPUS=cores.

Override the global configuration of ncpus computation

An EGO cluster administrator globally defines how ncpus is computed. The ncpus global definition can, however, be overridden on specified dynamic and static hosts in the cluster.

- Defining computation of ncpus on dynamic hosts
- Defining computation of ncpus on static hosts

Defining computation of ncpus on dynamic hosts

1. Open `ego.conf`.
    - Linux/UNIX: `$EGO_CONFDIR/ego.conf`
    - Windows: `%EGO_CONFDIR%\ego.conf`
2. Define the parameter EGO_LOCAL_RESOURCES="[**resource *resource_name***]".

    Set *resource_name* to one of the following:

    - define_ncpus_procs
    - define_ncpus_cores
    - define_ncpus_threads

    Note:

    Resource definitions are mutually exclusive. Choose only one resource definition per host.

    For example: EGO_LOCAL_RESOURCES="[resource define_ncpus_cores]"
3. Save and close `ego.conf`.

Note:

In multi-cluster environments, if ncpus is defined on a per-host basis (thereby overriding the global setting) the definition is applied to all clusters that the host is a part of. In contrast, globally defined ncpus settings only take effect within the cluster for which EGO_DEFINE_NCPUS is defined.

Defining computation of ncpus on static hosts

1. Open `ego.cluster.`*`cluster_name`*.
    - Linux/UNIX: `$EGO_CONFDIR/ego.cluster.`*`cluster_name`*
    - Windows: `%EGO_CONFDIR%\ego.cluster.`*`cluster_name`*
2. Find the host for which you want to define ncpus computation. In the `RESOURCES` column, add one of the following definitions:
    - define_ncpus_procs
    - define_ncpus_cores
    - define_ncpus_threads

    Note:

Resource definitions are mutually exclusive. Choose only one resource definition per host.

For example:

```
Begin Host
HOSTNAME  model  type    r1m  mem  swp  RESOURCES  #Keywords
#lemon    PC200  LINUX86  3.5  1    2    (linux)
#plum     !      NTX86    3.5  1    2    (nt)
Host_name !      NTX86    -    -    -    (define_ncpus.procs)
End Host
```

3. Save and close `ego.cluster.`*`cluster_name`*.
4. Restart the master host.

Note:

In multi-cluster environments, if ncpus is defined on a per-host basis (thereby overriding the global setting) the definition is applied to all clusters that the host is a part of. In contrast, globally defined ncpus settings only take effect within the cluster for which EGO_DEFINE_NCPUS is defined.

Performance tuning for large clusters

There are a number of system and Symphony parameters available that can be changed from their default settings to provide better performance for large, high-utilization clusters. These parameters are defined in configuration files and the Windows registry. As to what constitutes a large cluster, this is somewhat arbitrary. For the purposes of this discussion, we will consider a large cluster to have over 20,000 cores and over 100 applications. Since qualification of cluster size is not an exact science, relatively smaller clusters may also benefit to some degree from adjusting the parameters described here.

System configuration

Parameter	Registry Path/Configuration File	Description
EnableDynamicBacklog	HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\AFD\Parameters \EnableDynamicBacklog	Windows 2003 only. Enables or disables dynamic backlog. The default is 0 (disabled). Example: EnableDynamicBacklog=1 (REG_DWORD)
MinimumDynamicBacklog	HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\AFD\Parameters \MinimumDynamicBacklog	Windows 2003 only. Controls the minimum number of free connections allowed on a listening endpoint. If the number of free connections drops below this value, additional free connections are created. The default is 0. Example: MinimumDynamicBacklog=20 (REG_DWORD decimal)
MaximumDynamicBacklog	HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\AFD\Parameters \MaximumDynamicBacklog	Windows 2003 only. Controls the maximum number of free connections and connections in a half-connected (SYN_RECEIVED) state allowed on a listening endpoint. The default is 0 Example: MaximumDynamicBacklog=6144 (REG_DWORD decimal)
DynamicBacklogGrowthDelta	HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\AFD\Parameters \DynamicBacklogGrowthDelta	Windows 2003 only. Controls the number of free connections that are created when additional connections are necessary. The default is 0. Example: DynamicBacklogGrowthDelta=10 (REG_DWORD decimal)

Parameter	Registry Path/Configuration File	Description
MaxUserPort	HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\Tcpip\Parameters \MaxUserPort	Limits the number of dynamic ports available to TCP/IP applications. The default is 5000, which limits VEMKD support to 16K cores. Configuring MaxUserPort to 65534 enables VEMKD to support 20K cores. Example: MaxUserPort=65534 (REG_DWORD decimal)
TcpTimedWaitDelay	HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\Tcpip\Parameters \TcpTimedWaitDelay	Determines the time that must elapse before TCP can release a closed connection and reuse its resources. Example: TcpTimedWaitDelay=60 (REG_DWORD decimal)
tcp_window_scaling	/proc/sys/net/ipv4/ tcp_window_scaling	For RHEL5.3 x86_64 OS only. Set the value to 0.The default value is 1. With the default configuration, a client that is sending a large number of tasks (e.g. several hundred thousand) using message aggregation may experience a temporary "hanging" problem. When this problem happens, no tasks are submitted to Symphony from the client for about 15 minutes. After 15 minutes, the client API senses a broken connection with Symphony and automatically reconnects to Symphony and continues to send tasks. No user interaction is required. The problem will be automatically resolved as described above. This issue has only been noticed on RHEL5.3 x86_64 OS.

Symphony configuration

Parameter	Configuration File	Description
EGO_DISTRIBUTION_INTERVAL	ego.conf	Improves container startup speed, client response time, and overall cluster performance. The default is 5. Example: EGO_DISTRIBUTION_INTERVAL=1

Parameter	Configuration File	Description
EGO_ENABLE_COMPRESS_STATUS_FILE	ego.conf	Improves VEMKD file operation performance, thereby improving client response time and overall cluster performance. The default is "N". Example: EGO_ENABLE_COMPRESS_STATUS_FILE=Y
EGO_DYNAMIC_HOST_WAIT_TIME	ego.conf	Provides compute hosts with a greater chance to be recognized by the master host. The default is 60. Example: EGO_DYNAMIC_HOST_WAIT_TIME=60,120
EGO_DATA_MAXSIZE	ego.conf	Specified in Mbytes. Reduces the frequency for VEMKD to swtitch the stream file, to give the PERF loader sufficient time to finish loading data from the file to the database. Note: Adjust this parameter in conjunction with PERF egoeventsloader interval. The default is 10. Example: EGO_DATA_MAXSIZE=100
egoeventsloader->Interval	plc_ego.xml	Specified in seconds. Sets the interval shorter than the VEMKD stream file switch time so that the loader is able to load all data in the stream file before VEMKD switches it. Note: Adjust this parameter in conjunction with EGO_DATA_MAXSIZE in ego.conf. Example: <DataLoader Name="egoeventsloader" **Interval="300"** Enable="true" LoadXML="dataloader/egoevents.xml" />
EXINTERVAL	ego.cluster.<clustername>	Provides VEMKD with a greater chance to obtain load information from the master lim. Example: EXINTERVAL=150

Parameter	Configuration File	Description

MEM_HIGH_MARK and JAVA_OPT	wsm.conf	Reduces the chance of WEBGUI service restarts in a large cluster, and also helps with generating PERF reports.
		Example:
		MEM_HIGH_MARK=2048
		JAVA_OPTS="-Xms512m -Xmx2048m"

Java maximum heap size	EGO_TOP\perf\1.2.5\etc\plc.bat (Windows) EGO_TOP/perf/1.2.5/etc/plc.sh (Linux)	Gives more memory to the PLC service for collectiing data in a large cluster. Example: (Windows) "%JAVA_HOME%\bin\java.exe" -Xms64m **-Xmx2048m** ...
<ReclamationTimeout>	ConsumerTrees.xml	Reduces the chance of forcible reclaim happening. Example: <ReclamationTimeout>300</ReclamationTimeout>
EGO_MAX_CONN	ego.conf	Consider this only when there are more than 5000 physical hosts in a cluster. This configuration allows VEMKD to maintain more connections. Example: EGO_MAX_CONN=20000
SSM > startUpTimeout	Application profile	When there are many applications (e.g=. 300) in a cluster, SD may not be able to register all applications when many applications start SSM at the same time within the default SSM startUpTimeout. Consequently, some SSM processes may exit due to startup timeout. The default is 60 seconds. Increase this setting in a cluster with many applications. 300 seconds is suggested for clusters with 300 applications. Example: <SSM resReq="" workDir="${EGO_SHARED_TOP}/soam/work" **startUpTimeout="300"** shutDownTimeout="300">

PART

IV

Application Management

CHAPTER

17

Lifecycles

Application lifecycle

An application can be used when it is registered with the middleware. It can no longer be used when it is unregistered. There can only be one enabled application per consumer at any one time. You can only submit workload for an application that is enabled.

You register an application by registering the application profile with the Platform Management Console or the `soamreg command`.

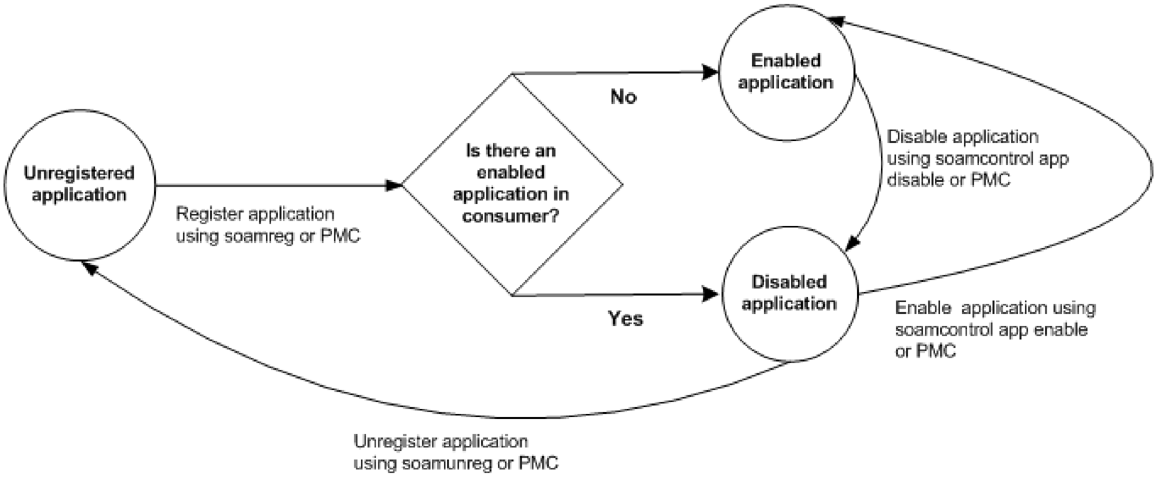
If there is already an enabled application in the same consumer, the application is registered but it is disabled.

If there are no enabled applications in the consumer, the application is registered and enabled.

Once you have registered an application, you can modify its parameters or update service packages through the Platform Management Console or the command-line.

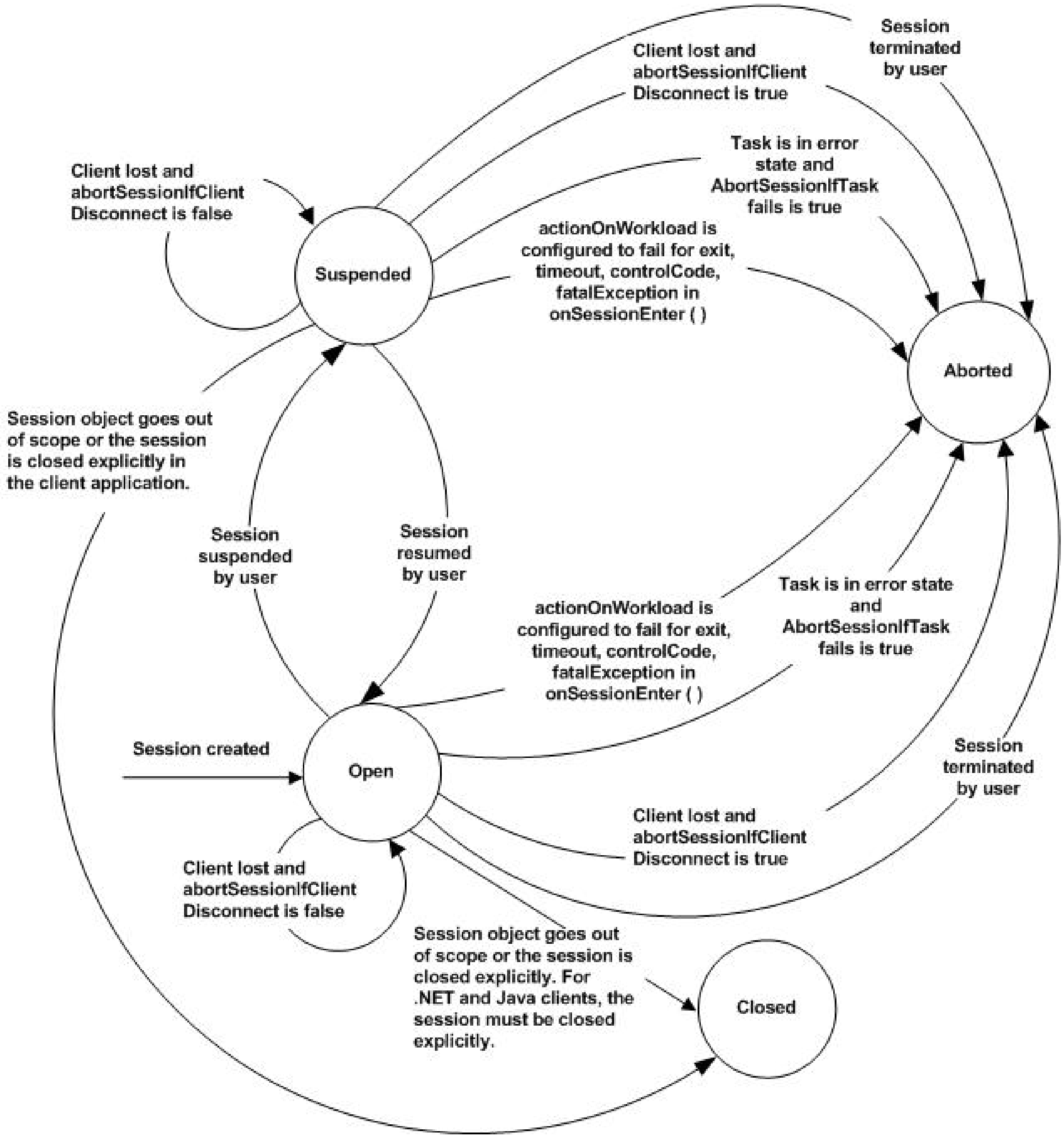
You disable an application with the Platform Management Console or the `soamcontrol app disable` command. When you disable an application, any workload for the application is terminated unless you choose to save workload when disabling it.

You remove an application from the system with the Platform Management Console or the `soamunreg command`. When you remove an application, existing sessions and tasks (running and suspended) are terminated, the application profile is deleted from the system, and all resources allocated to the application are released.


Enabled application
No
Unregistered application
Is there an enabled application in consumer?
Disable application using soamcontrol app disable or PMC
Register application using soamreg or PMC
Disabled application
Yes
Enable application using soamcontrol app enable or PMC
Unregister application using soamunreg or PMC

Session lifecycle

In your client application, you can create a local `Session` object that refers to a session in Symphony. You can interact with Symphony session by invoking methods on your local `Session` object.


Session terminated by user
Client lost and abortSessionIfClient Disconnect is true
Client lost and abortSessionIfClient Disconnect is false
Task is in error state and AbortSessionIfTask fails is true
Suspended
actionOnWorkload is configured to fail for exit, timeout, controlCode, fatalException in onSessionEnter ( )
Aborted
Session object goes out of scope or the session is closed explicitly in the client application.
Session suspended by user
Session resumed by user
actionOnWorkload is configured to fail for exit, timeout, controlCode, fatalException in onSessionEnter ( )
Task is in error state and AbortSessionIfTask fails is true
Session created
Open
Session terminated by user
Client lost and abortSessionIfClient Disconnect is true
Client lost and abortSessionIfClient Disconnect is false
Session object goes out of scope or the session is closed explicitly. For .NET and Java clients, the session must be closed explicitly.
Closed

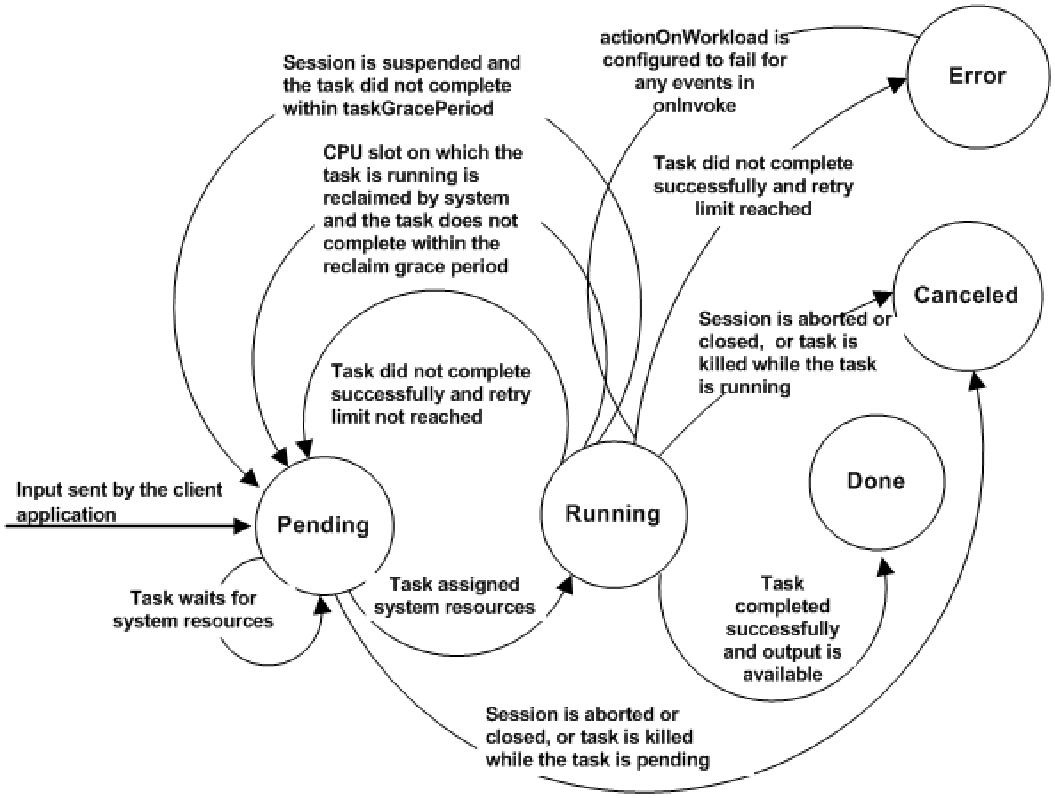
Task lifecycle

Once a task is created, it is valid in Symphony until one of the following conditions exist:

- The client collects the output and Symphony receives confirmation that the output was successfully collected
- The session is terminated, which in turn terminates all tasks
- The task is killed

You can specify that if a task fails it should be rerun by specifying the taskRetryLimit attribute in the application profile. This informs Symphony how many times it should rerun the task before giving up.

If a task fails, Symphony attempts to rerun the task up to the number of times specified by the taskRetryLimit attribute in the application profile. If the task has not successfully run after *n* retries, and the abortSessionIfTaskFail attribute is set true in the application profile then the session is aborted. Otherwise, a single task failure does not affect the session. By default, abortSessionIfTaskFail = false.

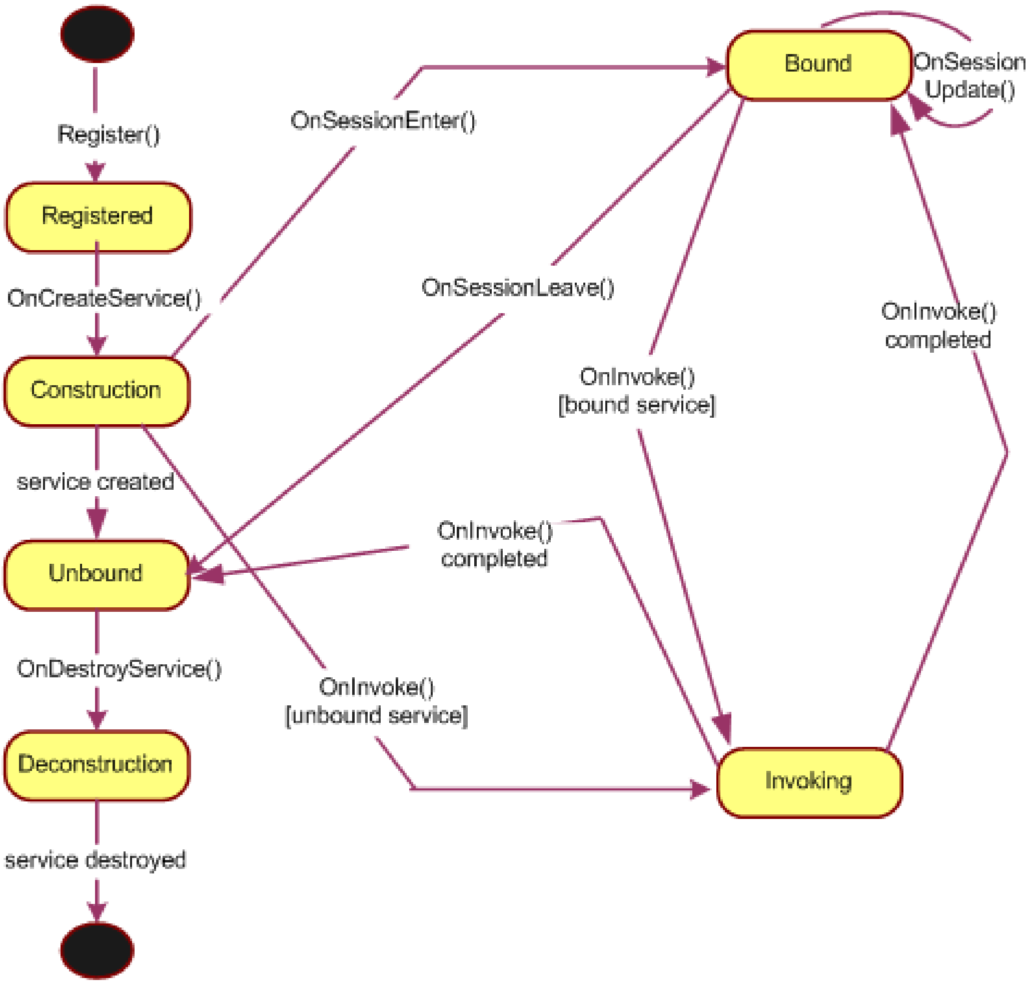


actionOnWorkload is configured to fail for any events in onInvoke
Error
Session is suspended and the task did not complete within taskGracePeriod
CPU slot on which the task is running is reclaimed by system and the task does not complete within the reclaim grace period
Task did not complete successfully and retry limit reached
Canceled
Session is aborted or closed, or task is killed while the task is running
Task did not complete successfully and retry limit not reached
Done
Input sent by the client application
Pending
Running
Task waits for system resources
Task assigned system resources
Task completed successfully and output is available
Session is aborted or closed, or task is killed while the task is pending

Service lifecycle

Symphony triggers state changes for the `ServiceContainer` as illustrated in the following diagram. The calls indicated on the diagram are calls made in the service, with the exception of Register() which is an internal call made by the system.

The arrows indicate a normal return of the method.



Note:

`OnSessionEnter( )` and `OnSessionLeave( )` are not called if the client does not relay common data when creating a session. `OnSessionUpdate( )` is not called if the client does not relay common data updates to the session.

Service instance lifecycle

A service instance is an executing instance of a service. There can be many instances of a service at any one time. Service instances are created by service instance managers.

Service instances can be started either before or when they are assigned to a session. They can stay running to compute multiple tasks of the same session to use the data and state information cached in memory for better performance. They can either exit or continue running when their serviced session is finished.

A service is transient if the service instances start and exit per session. A service is persistent if the service instances stay and serve multiple sessions. A persistent service is a long-running process like a daemon, which has to be more carefully programmed to avoid any accumulated problems such as memory leaks.

By default, the service instances in Symphony persist for multiple sessions. To make a service instance transient, you can return a control code from `onSessionLeave()` to tell Symphony to restart the service instance once it leaves the current session. This can also be used when you want to clean up any accumulated problems by restarting your service from time to time.

Timeouts that affect service instance life cycle

The service instance lifecycle can be affected by different configured timeouts. If an application has timeouts configured, then Symphony will take action if an operation exceeds the configured timeout. In this case, Symphony terminates the service instance, causing the cleanup methods not to execute under the following circumstances:

Method	**Not called**
onDestroyService()	• When an invocation of one of the following service methods times out: ◦ SessionEnter ◦ SessionUpdate ◦ Invoke ◦ SessionLeave • When a task cannot complete before the suspendGracePeriod expires. • When a task cannot complete before the taskCleanupPeriod expires. • When a resource on which the service instance is running is reclaimed, and the service instance cannot clean up before the applied reclaim grace period expires. • When the application is disabled or unregistered and the service instance cannot clean up before the cleanupTimeout expires. • When a middleware component becomes unavailable and the service instance cannot clean up before the cleanupTimeout expires.
onSessionLeave()	• When an invocation of one of the following service methods times out: ◦ SessionUpdate ◦ Invoke • When a task cannot complete before the suspendGracePeriod expires. • When a task cannot complete before the taskCleanupPeriod expires. • When a resource on which the service instance is running is reclaimed, and the service instance cannot clean up before the applied reclaim grace period expires. • When the application is disabled or unregistered and the service instance cannot clean up before the cleanupTimeout expires. • When a middleware component becomes unavailable and the service instance cannot clean up before the cleanupTimeout expires.

As a best practice, you should configure a timeout for the `onCreateService()` method if you believe that it may be hanging, otherwise some sessions may become under-allocated.

To verify if the method is hanging:

1. Run `soamview resource` to show that the resource is assigned to the session.
2. Run `soamview session -a` to show that the session deserves the resource but the resource is not assigned to it. Note that it is reasonable for the number or deserved resources and the number of assigned resources to be out-of-synch for some time (they will be out-of-synch while `onCreateService()` is executing); if their values do not seem to be converging after periodically monitoring them, you should suspect that the method is hanging.

CHAPTER

18

Deploying Applications and Files

Types of deployment

There are two types of deployment supported in Symphony:

1. Deployment of service packages, containing Symphony applications
2. File packages, which may contain files, binaries or other executables, patch updates, data or other files you want to make available to all or many hosts in the cluster

Both types of deployment deploy packages to the repository server, which stores the package files.

Service package deployment

Service package deployment is accomplished using the application deployment wizard, or the `soamdeploy` command. When you deploy a Symphony service package, the packages are downloaded to compute hosts on demand. This type of deployment is optimized to handle version updates of your service package.

This is the recommended method for deploying your Symphony service packages.

File package deployment

File package deployment is accomplished using the `rsdeploy` command. This type of deployment copies and installs one file package to many hosts at a time, whether the file package is needed by the receiving hosts or not. Install and uninstall operations to and from the compute hosts must be initiated by the user. There is no package version logic to manage versioning. Only hosts that are available at the time the installation operation occurs receive the package. All adding, removing, installing and uninstalling of packages are under direct user control.

This type of deployment can help you deploy Symphony patches more easily, load existing executables onto multiple hosts in the cluster, or load data required by many hosts.

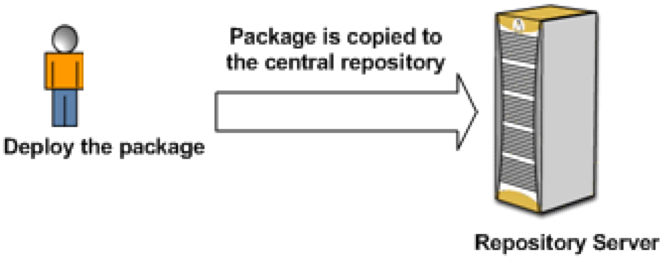
Service package deployment and removal process

The package deployment process has two phases: First, service packages are copied to the central repository on the repository server, the host on which the `rs` service is running. Then, when workload comes in, the service package is copied to compute hosts and uncompressed.

Package removal also has two phases: When a request to remove a package is made, service packages are removed from the central repository. Then, when a new application is deployed and existing packages on the compute hosts are no longer needed, packages are removed from compute hosts. For existing applications, when an existing package is updated, the packages that exist on compute hosts are overwritten when workload comes in.

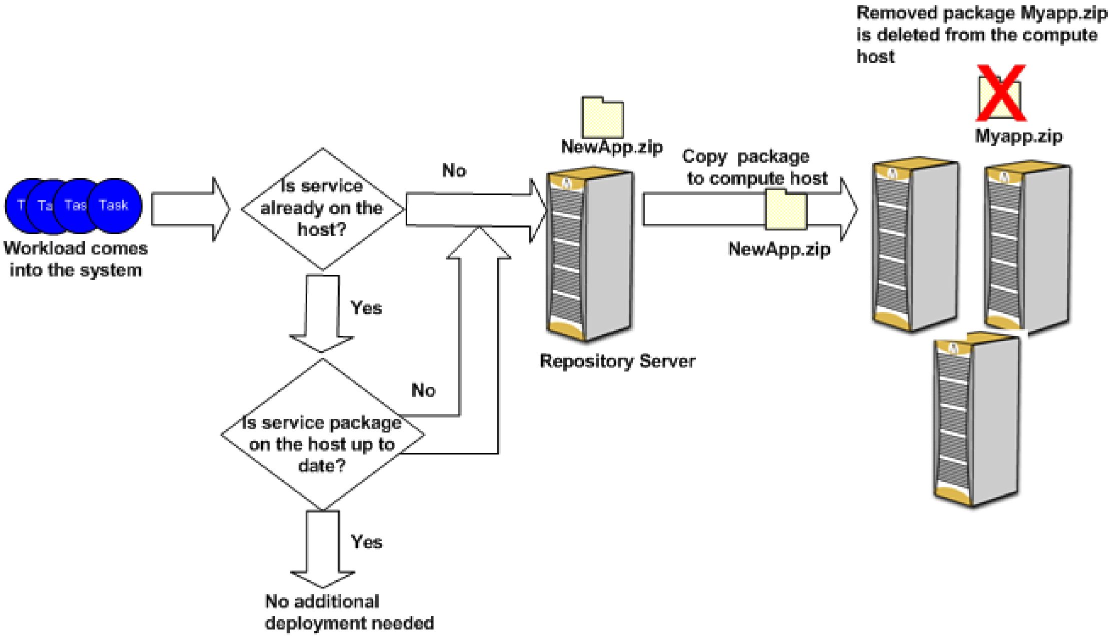
The package deployment process

1. You deploy the service package using the Platform Management Console or the `soamdeploy add` command. With the Platform Management Console, you use the Add/Remove Applications wizard or the global action Add package to repository in Manage Service Packages.
2. The package is copied to the repository server host.



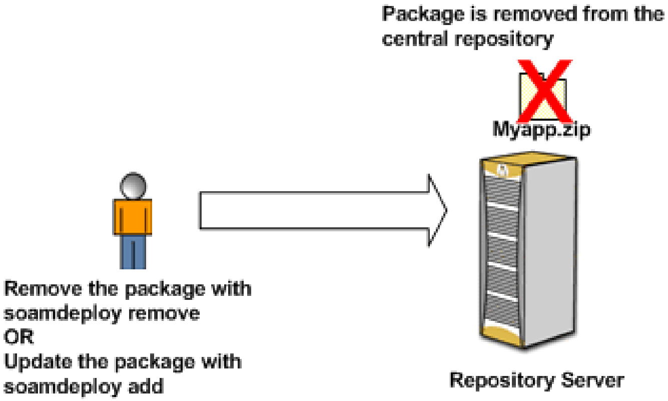
3. As workload comes in, the specified service in the application profile is requested for tasks. Platform Symphony checks whether the required service is already on the compute host.

   If the service is not already on the compute host, the Repository Service copies the service package from the repository server to the compute host, and uncompresses it, ready to be used.

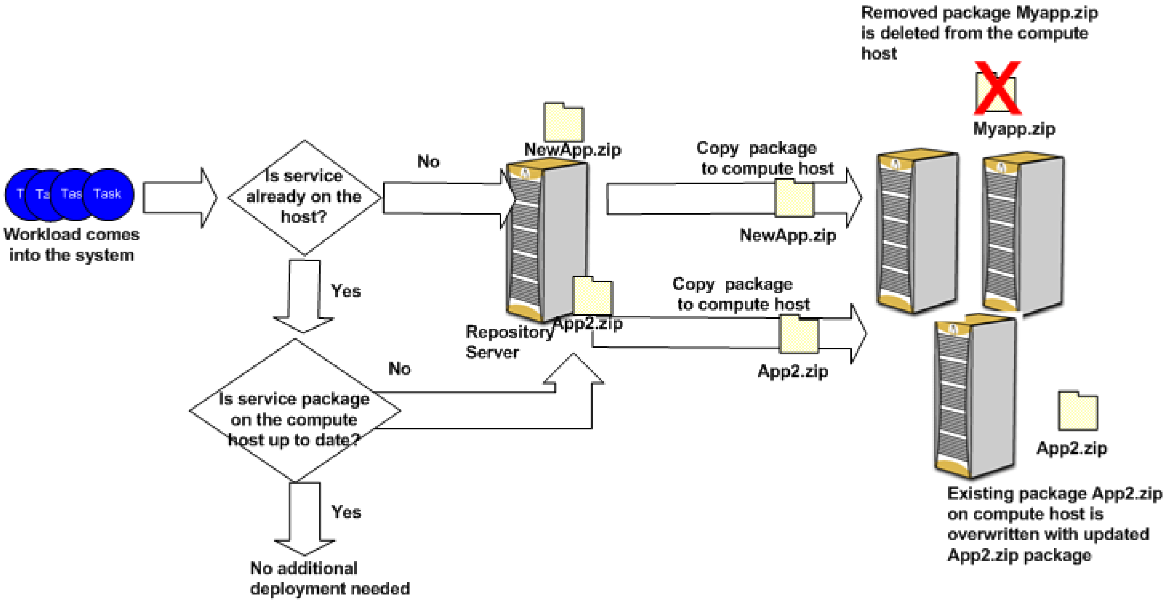

Removed package Myapp.zip is deleted from the compute host
Myapp.zip
NewApp.zip
Copy package to compute host
NewApp.zip
T Ta Tas Task
Workload comes into the system
Is service already on the host?
No
Yes
Repository Server
No
Is service package on the host up to date?
Yes
No additional deployment needed

The package removal process

1. You request to remove the service package using the Platform Management Console or the `soamdeploy remove` command, or you update an existing package through the Management Console or the `soamdeploy add` command.
2. The package is removed from the repository server host.


Package is removed from the central repository
Myapp.zip
Remove the package with soamdeploy remove
OR
Update the package with soamdeploy add
Repository Server

3. Whenever a new package is deployed on to the host, the removed package is deleted. Whenever an updated package is deployed on to the host, the existing package is overwritten with the updated package.


Removed package Myapp.zip is deleted from the compute host
Myapp.zip
T
Ta
Tas
Task
Workload comes into the system
Is service already on the host?
No
Yes
NewApp.zip
Repository Server
App2.zip
Copy package to compute host
NewApp.zip
Copy package to compute host
App2.zip
Is service package on the compute host up to date?
No
Yes
No additional deployment needed
App2.zip
Existing package App2.zip on compute host is overwritten with updated App2.zip package

Multiple repository servers

In a cluster that spans different geographic regions, the network connection between hosts in the different regions is typically a WAN. When a service in one region needs to download a large package from the repository server (RS), which happens to be in the other region, the connection can become a bottleneck and adversely affect other network traffic. To mitigate this issue, you can deploy a local RS. With this arrangement, a master RS retains global control of package deployment while each local RS serves an individual region. The local RS registers with the master RS when the local RS starts up. Each region can only have one local RS.

Cluster regions

The master RS maintains a table of which local RS should serve the package download request from a compute host. This table is defined in `$EGO_CONFDIR/../../eservice/rs/conf/region.xml`. There are two ways to configure the definition of a region:

1. You can define the region by IP range. The master RS checks the IP address of the compute host and redirects the request to the local RS.
2. You can also define the region by resource groups. The master RS checks the resource group the compute host belongs to and redirects the request appropriately.

The use of either IP ranges or resource groups must be consistent for all regions. i.e., you cannot mix the definition.

If the RS master is unable to find the region associated with a download request in the table, the master RS itself will service the request. Therefore, you do not need to configure the region where the master RS is running since the master RS will serve that region by default.

Region configuration

Here are some key details to be aware of when defining a region using IP ranges:

- Only IPv4 is supported
- Each field of the IP address can be:
  - a number (as 192)
  - a range (as 1-10)
  - a wildcard character (*)
- If one range is a subset of another range, or one range is overlapped by another range, the master RS selects the first region defined in `region.xml`.

Sample configuration of `region.xml` using IP range:

```
<Region name="region1">
<IPrange>172.168.1.1</IPrange>
<IPrange>192.168.2.1-100</IPrange>
<IPrange>192.168.4-10.*</IPrange>
</Region>
<Region name="region2">
<IPrange>10.10.*.* </IPrange>
</Region>
```

Sample configuration of `region.xml` using resource groups:

```
<Region name="region3">
<resourceGroup>rg1</resourceGroup>
<resourceGroup>rg2</resourceGroup>
</Region>
<Region name="region4">
<resourceGroup>rg3</resourceGroup>
</Region>
```

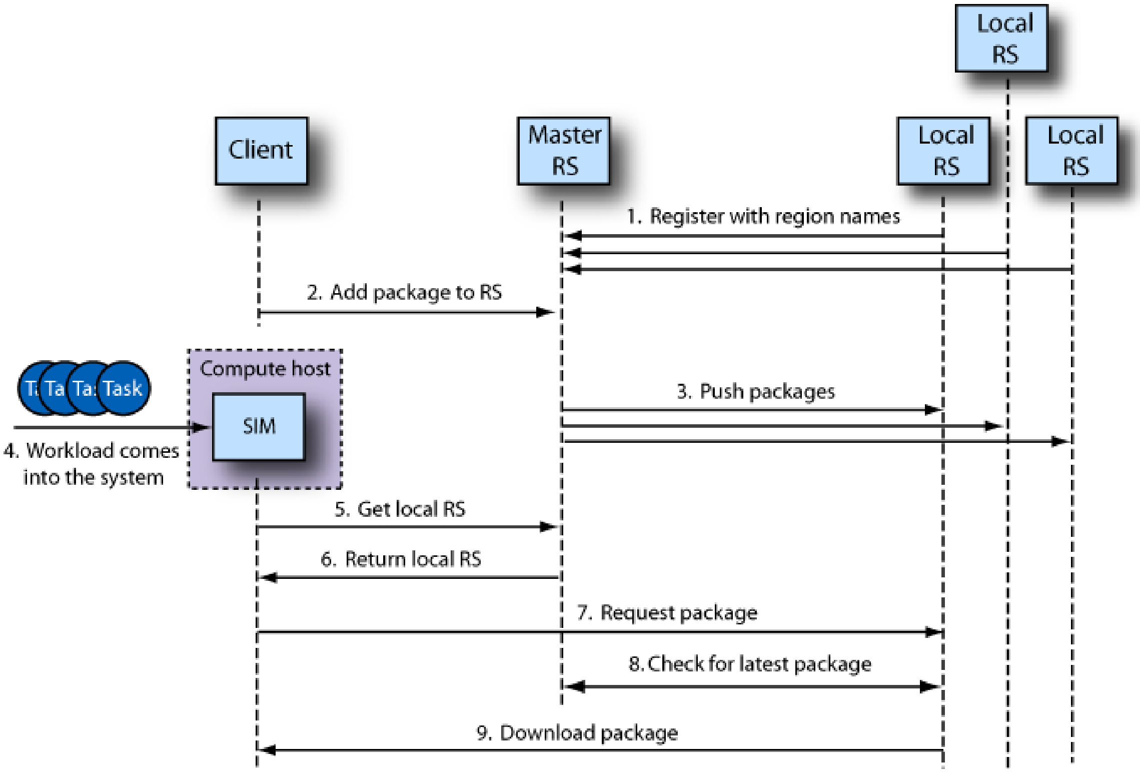
Local RS and master RS interaction

The local RS registers with the master RS when the local RS starts up. During registration, the local RS passes its region name to the master RS. If the master RS is down or the connection between the local RS and master RS is broken, the local RS will log an error message and try to re-establish a connection.

The master RS maintains a region/resource group table and a region/IP table. When a client needs to upload a package, it connects to the master RS. After the client uploads the package to the master RS, the master RS pushes the package to each local RS. If the local RS is down when the client uploads the package, the package in the local RS will not be in sync with the latest package. Therefore, the local RS always checks with the master RS for the latest version each time the package needs to be downloaded to a compute host.

When a compute host needs to download a package, it sends a redirection request to the master RS. (A redirection request basically asks the master RS for the URL of the local RS.) If the regions are defined using resource groups, the redirection request includes the compute host's resource group name. The master RS searches the `region.xml` file for the compute host's resource group or IP address, whichever is applicable, and redirects the request to the proper local RS.

When the client removes a package from the master RS, the master RS propagates the request to every local RS.



Local RS host requirements

The local RS host must meet the following requirements:

- It is accessible by all compute hosts in the local region.
- It has enough disk space (either local or shared) to store all packages.
- The OS currently supports RS.

Configure regions according to IP range

Perform this task when you want to configure the geographic regions of a cluster and there are resource groups in the cluster that spans multiple regions.

1. From the PMC, select Resources > Repository Service Regions.
2. In the dropdown list next to Define regions by, select IP Address.
3. In the Global Actions dropdown list, select Create a new Region.

   The Create A New Repository Service Region dialog displays.
4. Enter the name of the region. Names must be unique for each region.
5. Enter the IP range in the fields provided that will be associated with the local RS. If you want to specify multiple IP ranges, click Add and enter the range. Note that the following values are valid values for the IP address fields: number, number-number, or *.
6. Do one of the following:
   - If you want the local RS to be started on a host in an existing resource group, choose Existing resource group and select the resource group from the dropdown list.
   - If you want the local RS to be started on a host in a new resource group, select Create new resource group ... and enter the RS host name in the textbox. If you want to specify additional hosts for failover, separate them with a (,).
7. If necessary, change the listening port number of the new local RS.
8. Click in the Specify repository directory ... checkbox if you want to set up a shared directory for the local RS.
9. Click Create.

Configure regions according to resource groups

Perform this task when you want to configure the geographic regions of a cluster and there is no resource group in the cluster that spans multiple regions.

1. From the PMC, select Resources > Repository Service Regions.
2. In the dropdown list next to Define regions by, select Resource Group.
3. In the Global Actions dropdown list, select Create a new Region.

   The Create A New Repository Service Region dialog displays.
4. Enter the name of the region. Names must be unique for each region.
5. Select resource groups that belong to the region from the Available resource groups section.
6. Do one of the following:
   - If you want the local RS to be started on a host in an existing resource group, choose Existing resource group and select the resource group from the dropdown list.
   - If you want the local RS to be started on a host in a new resource group, choose Create new resource group ... and enter the RS host name in the textbox. If you want to specify additional hosts for failover, separate them with a (,).
7. If necessary, change the listening port number of the new local RS.
8. Click in the Specify repository directory ... checkbox if you want to set up a shared directory for the local RS.
9. Click Create.

Managing service package dependencies

Some service packages depend on common libraries that may require periodic updates. For these service packages, it is possible to separate common libraries into individual packages and maintain the dependencies so that the service package only includes the main binaries.

With this feature, a complex package can be organized into multiple smaller packages, which can be more easily maintained. It also saves disk space in the RS server as well as reducing the SIM download time if several packages share common libraries. A package that has a dependency can only depend on packages that are deployed at the same consumer or its ancestor consumers. The packages that a main service binary depends on are known as dependent packages. In a typical case, the dependent package libraries are deployed under root consumer "/". Packages deployed with short names can only depend on packages deployed under the same consumer or root consumer. Packages cannot have mutual dependencies among them, e.g., both package A and package B are deployed under the same consumer; A depends on B, and B depends on A.

Package upload

A package can be uploaded/updated individually even if it has dependency on other packages or other packages have dependency on it. The RS performs a dependency check when a package is uploaded and gives a warning if the dependent packages are not deployed yet. It is recommended to upload the dependent pacakges first to avoid such warnings.

Package removal

When removing packages with dependencies, it is recommended to remove the packages that have no other packages depending on them first, then remove the dependent packages. Otherwise, the dependency check fails and the RS rejects the remove request.

Package dependency and dynamic application updates

If any package of an application changes, whether it is the package the application directly uses or the packages it indirectly depends on, a dynamic application update is triggered. Following the change, the updated package will be downloaded by the SIM in the compute host.

Package dependency and application profiles

Only the package that is directly used by an application is configured in the application profile. The dependent packages do not appear in the application profile and are only maintained by the RS. This way, the application profile is not affected by any dependency changes.

Internally, the SD sends an application profile to the SSM that includes package version as well as all dependent packages. The SIM downloads the packages and all its dependencies by reading the application profile sent from the SSM. The SIM cannot start a task unless all packages have been successfully downloaded and are current.

Package environment variables

You can define environment variables for each package that are propagated to the service instance. The definition can use any previously defined environment variable and "SOAM_DEPLOY_DIR". Here "previously defined variables" includes variables defined in package dependencies. Example:

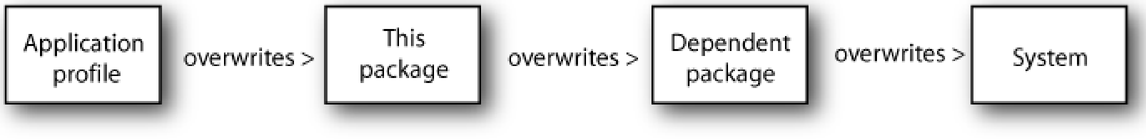
JAVA_HOME=${SOAM_DEPLOY_DIR}

PATH=${JAVA_HOME}/bin:${PATH}

On Windows, ${SOAM_DEPLOY_DIR} of each dependent package is added to the PATH environment variable for the service. On Linux/UNIX, ${SOAM_DEPLOY_DIR} of each dependent package is added to the PATH and LD_LIBRARY_PATH environment variables for the service.

Here is the order of precedence for environment variables used by services:

- If the environment variable defined in the package or any package dependencies already exists in the system, it will overwrite the one in the system.
- If a dependent package defines an environment variable that is the same as the one defined in this package, the one defined in this package overwrites the environment variable in the package dependency.
- If two dependent packages define the same environment variable, the package listed later in the deployment.xml overwrites the former.
- If the same environment variable is defined in one package more than once, the one defined later is ignored.



Configure service package properties

Perform this task when you want to configure service package dependencies, environment variables, or commands to run when installing/uninstalling the service package.

1. From the PMC, select Symphony Workload > Manage Service Packages > Packages.
2. From the list of service packages, click the name of the package for which you want to configure dependencies and environment variables.

   The Service Package Properties dialog displays.
3. Under the Dependent Packages heading, click the Add/Remove Dependency button.
4. Select the package(s) that the service requires to run properly.

   Note:

   If the dependent package does not appear in the list, verify that it has been added to the service repository.
5. If you want to want add an environment variable that will be accessible to the service, expand the Environment Variables section and enter a variable name and value.
6. If you want to use a third-party program that needs to be installed on a compute host, you can configure it in the install/uninstall sections of the Service Package Properties dialog.
7. Click Finish.

Deploying a new application

Goal

You developed a new service, compiled it, and are now ready to use it in your cluster. To use the new service, you need to deploy it to compute hosts and associate it with an application.

At a glance

1. Create the service package
2. Create the application
3. Configure the client to run with the new application

Create the service package

Before you can deploy a service, you need to create a service package.

Packaging a Symphony application for deployment involves putting all service files and any dependent files associated with the service in a package.

Important:

Verify that all dependencies are either pre-installed or deployed with the service. For example, if your application is .NET, ensure that the .NET Framework is installed and that it is the correct version for your application.

Compress into a package:

- Service executables
- Additional files required for the services to work.

It is not required to use gzip as indicated in the example to package a service. You can use any supported format. If using a utility other than gzip, ensure the compression and uncompression utility is in your Path environment variable when using soamdeploy.

Supported package formats:

- .zip
- .tar
- .taz
- .tar.zip
- .tar.Z
- .tar.gz
- .tgz
- .jar
- .gz

Create the service package for deployment on Windows

1. Go to the directory in which the service is located.

   For example, `%SOAM_HOME%\5.1\win32-vc7\samples`
2. Create an application package by compressing the service executable into a zip file:

   gzip SampleService.exe

You have now created your first service package `SampleService.exe.gz`. Next, create the application.

Create the service package for deployment on Linux/UNIX

1. Go to the directory in which the service is located.

   For example, `$SOAM_HOME/5.1/linux2.4-glibc2.2-x86/samples`
2. Create an application package by compressing the service executable into a tar file:

   tar -cvf SampleService.tar SampleService

   gzip SampleService.tar

   You have now created your first service package `SampleService.tar`.gz. Next, create the application.

Create the application

Add the application with the Add Application Wizard. The Wizard creates a consumer for you with your application name, allocates resources, deploys your service package and registers your application. After completing the Wizard, your application should be ready to use.

You can also use commands to deploy a service package and register an application. See `soamdeploy` and `soamreg` in the *Symphony Reference* for more details.

1. In the Platform Management Console, click Symphony Workload > Configure Applications.

   The Applications page displays.
2. Select Global Actions > Add/Remove Applications.

   The Add/Remove Application page displays.
3. Select Add an application, then click Continue.

   The Adding an Application page displays.
4. Select Create new profile and add application wizard.
5. Enter your application name, then click Continue.

   The Define the Service window displays.
6. Enter service information, then click Continue.
   a) Change the Service name to the name you want to assign to your new service.
   b) In command to start this service, enter the command to run your service executable.

      For example, if in your service package you have the directory structure `\myservice\myservice.exe`, indicate `myservice\myservice.exe` in StartCmd.

   The Define session type window displays.
7. If you have defined session types in your client application, select Define a custom session type, fill in the desired information, then select Continue.

   The Confirm application profile details window displays.
8. Review your selections, then click Confirm.

   The window displays indicating progress. Your application is ready to use.
9. Click Close.

The window closes and you are now back in the Platform Management Console. Your new application is displayed as enabled.

Deploy a service package using the PMC

To use a new service, you must deploy the service binary to your cluster. Using the Platform Management Console (PMC) to update an existing package will not terminate the current running workload.

You can deploy the service package to a non-leaf consumer so that all applications registered to child leaf consumers are able to share the same service package. To deploy a service to a non-leaf consumer, you must ensure that:

- you are a consumer admin for the level at which the package is being deployed, or your user account is assigned an RBAC role that contains "Deploy package" permission and the associated consumer is this non-leaf consumer or higher level consumer.
- the name of the service package is unique within the branch, i.e. , you cannot have a leaf inheriting two packages with the same name.

Deploy a service package

1. In the Platform Management Console, select Symphony Workload > Manage Service Packages.
2. In the consumer tree, select the consumer that you want to associate with the service package.
3. In the Global Actions dropdown list, select Add Package to Repository.

   The Add Package to respository page displays.
4. Browse to your service package and select it. Click Add.
5. Select the application associated with your service package, then Add.

   Your service package should now be displayed in the list.

Deploy a service package using the CLI

To use a new service, you must deploy the service binary to your cluster. Using the `soamdeploy add` command to update an existing package will not terminate the current running workload.

You can deploy the service package to a non-leaf consumer so that all applications registered to child leaf consumers are able to share the same service package. To deploy a service to a non-leaf consumer, you must ensure that:

- you are a consumer admin for the level at which the package is being deployed, or your user account is assigned an RBAC role that contains "Deploy package" permission and the associated consumer is this non-leaf consumer or higher level consumer.
- the name of the service package is unique within the branch, i.e. , you cannot have a leaf inheriting two packages with the same name.

Deploy a Windows service package

Verify that all dependencies are either pre-installed or deployed with the service. For example, if your application is .NET, ensure that the .NET Framework is installed and that it is the correct version for your application.

To use a new application, you must deploy the binary to hosts in your cluster.

1. Deploy the service package with the `soamdeploy` command:

   soamdeploy add SampleService -p SampleService.exe.gz -c /SampleApplications/SOASamples

   The service package is deployed.
2. Check the list of deployed services with the `soamdeploy view` command:

   soamdeploy view -c /SampleApplications/SOASamples

Deploy a Linux/UNIX service package

1. Deploy the service package with the `soamdeploy` command.

   soamdeploy add SampleService -p SampleService.tar.gz -c /SampleApplications/SOASamples

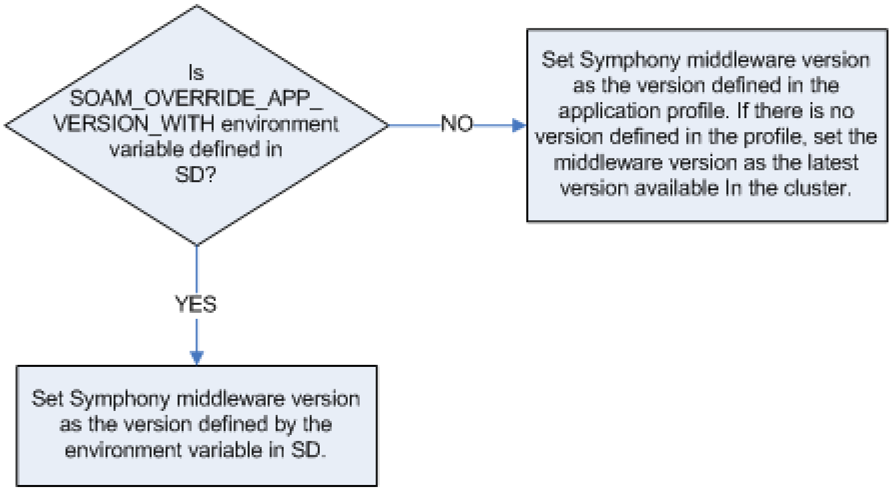
   The service package is deployed.
2. Check the list of deployed services with the `soamdeploy view` command:

   soamdeploy view -c /SampleApplications/SOASamples

Overriding application profile versions cluster-wide

You can instruct the Symphony middleware to override the configured version of all application profiles currently registered in the system with another version and to run workload on that other version.

When the application version override is configured, the SD must be restarted for the change to take effect, The next time the SD starts an application, it substitutes the value of the `version` attribute in the application profile with the version configured globally before passing the profile to the SSM. The version override feature is enabled using the SOAM_OVERRIDE_APP_VERSION_WITH environment variable in the SD. The following diagram summarizes the behavior of this feature.


Is SOAM_OVERRIDE_APP_VERSION_WITH environment variable defined in SD?
NO
Set Symphony middleware version as the version defined in the application profile. If there is no version defined in the profile, set the middleware version as the latest version available In the cluster.
YES
Set Symphony middleware version as the version defined by the environment variable in SD.

This global setting takes effect on all profiles and any specific versions specified in the application profile at run time are ignored.

This setting does not have any effect during the registration of a profile, i.e., the validation logic applied during the registration of a profile is still done with respect to the version configured in the profile; for example, if the global setting is version 5.0, a profile containing a version of 4.0 is still validated against a version 4.0 schema when it registered. However, the application will run on the version 5.0 middleware.

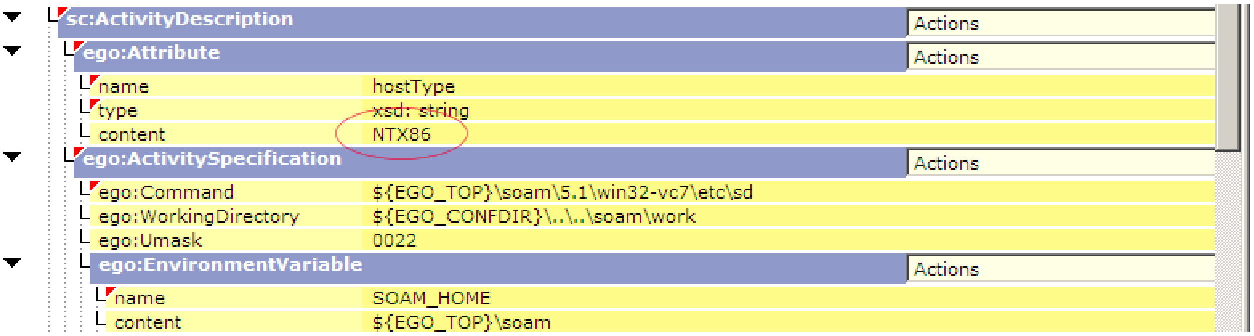
A key fact to be aware of when using this feature is that the SD does not change the profile on the disk when overriding the application version. This means that both the CLI and the PMC application editor will show the application profile as it is on disk, which may be different than the running instance of the application after a version override has occurred.

Application version override configuration

To allow the middleware versions of all applications in the cluster to be overridden without any change to the application profiles, you must configure the following parameter in SD.xml.

1. From the PMC, select System Services > Configure Services.
2. Click SD.

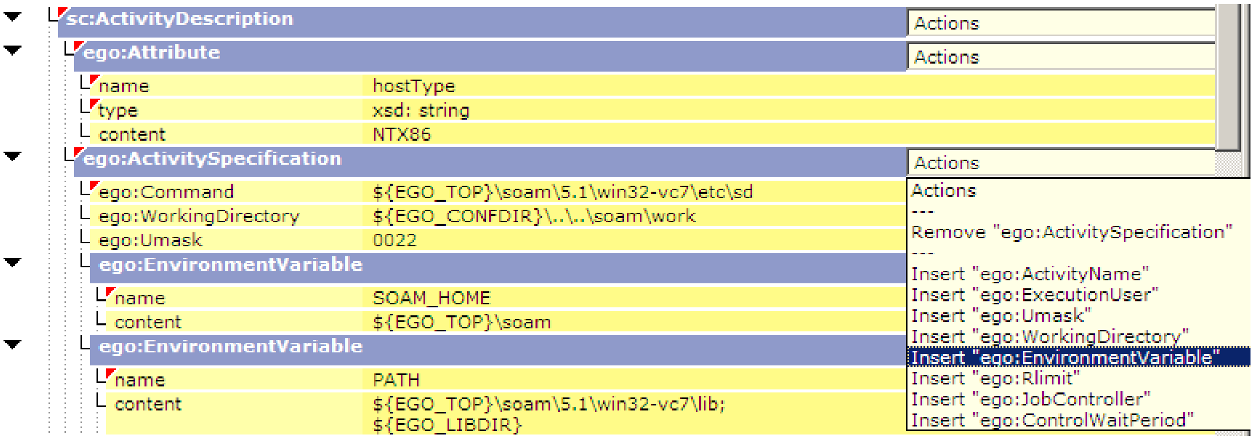
   The SD servfice profile displays.
3. Locate the sc::ActivityDescription parameter for the appropriate host type; for example, NTX86. The host type is specified by the content attribute in ego:Attribute.



```
sc:ActivityDescription
 ego:Attribute
  name                    hostType
  type                    xsd: string
  content                 NTX86
 ego:ActivitySpecification
  ego:Command             ${EGO_TOP}\soam\5.1\win32-vc7\etc\sd
  ego:WorkingDirectory    ${EGO_CONFDIR}\..\..\soam\work
  ego:Umask               0022
  ego:EnvironmentVariable
   name                   SOAM_HOME
   content                ${EGO_TOP}\soam
```

4. In the Actions drop-down list of the ego:ActivitySpecification parameter, select Insert "ego:EnvironmentVariable".

   A new ego:EnvironmentVariable parameter is added to the profile.


Actions
Actions
Actions
Actions

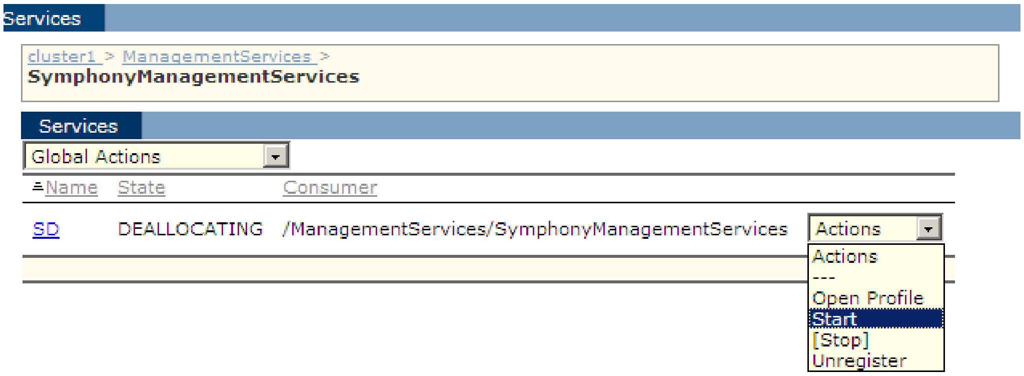
Remove "ego:ActivitySpecification"

Insert "ego:ActivityName"
Insert "ego:ExecutionUser"
Insert "ego:Umask"
Insert "ego:WorkingDirectory"
Insert "ego:EnvironmentVariable"
Insert "ego:Rlimit"
Insert "ego:JobController"
Insert "ego:ControlWaitPeriod"

```
sc:ActivityDescription
 ego:Attribute
  name                    hostType
  type                    xsd: string
  content                 NTX86
 ego:ActivitySpecification
  ego:Command             ${EGO_TOP}\soam\5.1\win32-vc7\etc\sd
  ego:WorkingDirectory    ${EGO_CONFDIR}\..\..\soam\work
  ego:Umask               0022
  ego:EnvironmentVariable
   name                   SOAM_HOME
   content                ${EGO_TOP}\soam
  ego:EnvironmentVariable
   name                   PATH
   content                ${EGO_TOP}\soam\5.1\win32-vc7\lib;
                          ${EGO_LIBDIR}
```

5. In the name field, enter SOAM_OVERRIDE_APP_VERSION_WITH.
6. In the content field, enter the SOAM version .
7. Click Save. Click OK.

   A successful update message displays.
8. Click OK.
9. In the Actions drop-down list for the SD service, select Start.



Name	State	Consumer
SD	DEALLOCATING	/ManagementServices/SymphonyManagementServices

The SD service is restarted. The application version change will take effect when the applications are started.

Enable an application to always use the latest Symphony version

If you want to ensure that an application always runs on the latest version of Symphony middleware available in the cluster, you should not configure a version in the application profile. The absence of a version is interpreted by Symphony as "run on latest version" in which the latest version is assumed to be the same version as the SD. A version of Symphony is considered to be available if it is the only version in the cluster or, in the case of multiple Symphony versions in the cluster, it was at some time activated during a direct upgrade.

When you configure an application to use the latest version, it is assumed that the registered application profile contains enough application configuration to allow it to be used by the latest version of Symphony, i.e., no change is required to the application profile unless you intend to make use of a feature that was introduced in the newer version of Symphony.

When the SD encounters these types of profiles, it will override the `version` variable in the profile before passing it to the SSM.

It is recommended to use this feature if you want to remove control from the application over which version of Symphony it runs its workload on when the cluster is upgraded.

Here are some examples of when this feature could be used:

- You install a newer version of Symphony on a separate cluster and register an older profile and want workload to run on the new version with no change to the profile.
- You have done a direct upgrade on your existing version of Symphony and want application workload to run on the new version with no change to the current profile.
- You have done a direct upgrade on your existing version of Symphony and want only specific applications that do not specify the profile version to run workload on the new version with no change to the profile. All other applications will continue to run on the version specified by their respective application profiles.

Configuring an application to always use the latest Symphony version

Follow these steps when you want to update an existing application so that it always runs on the latest version of middleware available in the cluster.

1. From the PMC, select Symphony Workload > Configure Applications.
2. Click the application name.

   The application profile displays.
3. Select either Basic Configuration or Advanced Configuration from the drop-down list.
4. Under SOAM Version, select Always use latest available version.
5. Modify any other profile settings, as required

Register a new application

You must create at least one consumer, and deploy the service package to the consumer before you can register an application.

You can register more than one application per consumer but you can have only one enabled application per consumer at any given time.

You can only register an application at a leaf consumer (a consumer that has no sub-consumers).

Tip:

As an alternative, you can use the Add/Remove Application wizard, a tool that walks you through all the steps required to successfully add an application to your cluster. Access the wizard from the dashboard in the gird, or from the Symphony Workload page, **Configure Applications** > **Add/Remove Application**.

1. Click Symphony Workload > Configure Applications.
2. On the tree, click the leaf consumer name (the lowest level you created) for which you want to register a new application.

   A list of enabled and disabled applications for that consumer displays.
3. Select Global Actions > Add Application using the profile editor.

   The Register a new application window displays.
4. From the drop down list, select Basic Configuration or Advanced Configuration.
5. Fill in the values or click Import... if you want to modify an existing application.

   Note:

   If you import the application profile and the version specified in the profile is not present in the cluster , the editor will set the version of the imported application to always run on the latest version.
6. Click Register.

Minimizing application changes by using short consumer names

Consumers are organized in a tree structure that models the organizational relationships among consumers. A short consumer name is defined as the name of a consumer without any path information of the tree. A full consumer name is defined as the name of a consumer with complete path information.

When a full consumer name is used in an application profile or service deployment and you need to modify the consumer tree hierarchy, the following components affected by the change must be updated in order to keep the applications working:

- the application profile must be updated to use the new full name
- the service packages must be re-deployed under the new full name. The existing package under the original full name must be deleted
- borrow/lend relationships defined in the ownership policy must be reconfigured

The benefit of the full consumer name is that it enables Symphony to do more error checking to keep cluster configuration consistent. The drawback is that when the consumer tree hierarchy changes, more effort is required to update the application and borrow/lend relationships.

The benefit of the short consumer name is that when the consumer tree hierarchy changes, there is no need to re-deploy service packages or re-register application profiles. This benefit minimizes the effort to maintain and administer applications and service packages.

Consumer name guidelines

You can specify either a short consumer name or full consumer name to identify a consumer when you perform the following operations:

- deploy or access a package
- register or manage an application profile
- administer an application using CLI or PMC

Full names and short names can co-exist in the cluster provided they are for different consumers.

The following guidelines apply when using short and full consumer names. For the sake of brevity, short and full consumer names are referred to as simply short names and full names, respectively.

Deploying and accessing packages

- For each leaf consumer, you can deploy service packages using either the full name or short name. You cannot use any other format, such as a relative path, in the consumer name
- If a service package is deployed with a full name, the application profile must also specify the full name in order to use it.
- When a service package is deployed under "/", it can be accessed by any consumer no matter if the application profile specifies the consumer with a short name or full name. If the package exists at the root consumer "/", the package cannot be deployed to any other consumer, regardless of whether it is deployed with a short consumer name or full path.
- When a service package is deployed under a non-leaf node other than "/", it can only be used by an application if its application profile uses a full name. The reason is to make sure applications get the correct packages when the consumers are moved around in the tree. If the package is also deployed under a short name of a consumer which is the child of the non-leaf node above, the one deployed with a short name will be used no matter if the application specifies a short name or full name.

- It is not allowed to deploy a service package under a short name when there exists one that was deployed under a full name. This is safeguard to prevent errors that can cause the workload to stop running. For the same reason, it is also not allowed to deploy a package under a full name when there exists one that was deployed under a short name. In both cases, if you really want to change the type of consumer name, the old package must be explicitly removed before it can be re-deployed.
- Security checks, which are performed when accessing a service package or re-deploying a service package, are based on the current consumer tree, not the tree when the package was first deployed.

  Example:

  T1: consumer A's full name was /test/A. You need to have administrator privilege for /test/A to deploy the package

  T2: consumer A's full name changes to /production/A. If you want to remove or re-deploy the package, you need to have administrator privilege for /production/A.

Register or managing an application profile

- If a service package is deployed with a full name, the application profile must also specify the full name in order to use it.
- If a service package is deployed with a short name, the application profile can either specify a full name or short name. In the case where the application profile is using a full name, if the consumer tree hierarchy changes, it is not necessary to re-deploy the package but you need to change the application profile.

Administering an application using CLI or PMC

- Whether the PMC shows a service package with a short name or full name depends on which one was used when it was deployed.
- When the PMC shows runtime information (for example, the application properties page), it will always show the full name no matter how it was configured. However, for service package management, the consumer name is whatever was used when it was deployed.
- If the service package is deployed with a short name, the CLI soamdeploy command can access the package by short name only. Similarly, if the package is deployed with a full name, the soamdeploy command can access the package by full name only.

Using a short consumer name when deploying a service package with the PMC

Perform these steps when using a short consumer name in association with a service package that you want to add to the repository.

1. From the PMC, select Symphony Workload > Manage Service Packages.
2. Select the relevant consumer.
3. Select Global Actions > Add package to repository.
4. Click in the deploy use short consumer name checkbox. Continue the configuration as described in the "About service packages" help topic.

Using a short consumer name when registering a new application profile with the PMC

Perform these steps when you want to configure a short consumer name in the application profile.

1. From the PMC, select Symphony Workload > Configure Applications.

2. On the tree, click the leaf consumer name (the lowest level you created) for which you want to register the application.

   A list of enabled and disabled applications for that consumer displays.
3. Select Global Actions > Add Application using the profile editor.

   The Register a new application window displays.
4. From the drop down list, select Advanced Configuration.
5. Click in the use short name for consumer check box.
6. Fill in the remaining values.
7. Click Register.

Deploy a service package with your own deployment tool

You do not want to use the deployment tool distributed with Symphony to deploy your service packages. You have your own tool but want it to work with Symphony. You do not need to create a service package to use your own deployment tool.

- If the service binaries are in a shared location, service binaries and any other additional files required by the service must be accessible to all compute hosts
- If service executables are locally installed on compute hosts, the service executables must be in the same location on all compute hosts
- Symphony grid only. The OS user account assigned to the consumer for the application must have permissions to execute the service binaries on compute hosts

1. In the Platform Management Console, create an application with the Add/Remove Application wizard.
2. Click Configure Applications and select the application to modify.

   The Application Profile window displays.
3. Click Export and save it to a file.
4. In an XML editor, open the application profile and edit the Service section:
   a) For `PreExecCmd` , specify the command to run for your deployment tool to deploy your service on to the compute hosts.
   b) For `StartCmd` , specify the location of the service binary on the compute host after the deployment command has run. This location must be the same on all compute hosts.
   c) In `workDir`, specify the working directory for your service.

      On Windows:

      For example, if your deployment command is called `deploy`:

```
<Service name="myservice" description="My Sample Service">
...
<osType name="all" preExecCmd="C:\mydeploytool\bin\deploy.exe download -a myservice" startCmd="C:
\myservices\myservice\myservice.exe"
workDir="C:\myservices\myservice\work">
</osType>
...
</Service>
```

If working directory is not specified, by default, `%SOAM_HOME%\work` is used for service instances.

On Linux/UNIX:

```
<Service name="myservice" description="My Sample Service">
...
<osType name="all" preExecCmd="/mydeploytool/bin/deploy download -a myService" startCmd="/
myservices/myservice/myservice" workDir="/myservices/myservice/work">
</osType>
...
</Service>
```

5. Click Configure Applications and select the application to modify.

   The Application Profile window displays.
6. Click Import and browse to select the changed application profile, then click Import.

Note:

You can also use the soamreg command to register your new application profile.

7. Click Save to save your changes.

Deploy a service package without a deployment tool

You do not need to use the deployment tool distributed with Symphony to deploy your service packages. You want to skip the deployment step altogether, and still be able to tell Symphony where the service binaries are located on each machine.You may need to do this because you image your compute hosts with the service binaries already on the machines, or service binaries are in a shared location, so deployment is not necessary.

- You do not need to create a service package—you can copy the service binaries and any additional required files to the desired location
- If the service binaries are in a shared location, service binaries and any other additional files required by the service must be accessible to all compute hosts
- If the service binaries are locally installed on compute hosts, the service binaries must be in the same location on all compute hosts
- Symphony grid only. The operating system user account assigned to the consumer for the application must have permissions to execute the service binaries on compute hosts

1. Copy the service binaries to the desired location.
2. In the Platform Management Console, create an application with the Add Application Wizard.
3. Click Configure Applications and select the application to modify.

   The Application Profile window displays.
4. Under Service Definition, Operating System Definition, change the Start Command and Work Directory.
   a) Start Command—Specify the path to your service binary.

      On Windows:

      For example, if the service1 binary is located locally on each machine in `c:\myservice\service1.exe`, specify

      ```
      C:\myservice\service1.exe
      ```
      .

      On Linux/UNIX:

      For example, if the service1 binary is located locally on each machine in `/share/myservice/service1`, specify: **/share/myservice/service1**

   b) Work Directory—Specify the absolute path on the compute host to the directory in which the service creates files.

      On Windows, for example: `C:\myservice\work`.

      On Linux/UNIX, for example: `/share/myservice/work`.
5. Click Save to save your changes and update your application profile.

Automatically run a command when deploying a service package

Suppose your service uses a third-party tool and it needs to be installed on the compute host, or you want to run a script to perform some actions for proper functioning of the service program. You can configure this in a package-specific `deployment.xml` configuration file.

Windows service package

1. Create a file for your service package with the name `deployment.xml`.

   The file must be called `deployment.xml`.

   For example:

```
<Deployment xmlns="http://www.platform.com/Symphony/Deployment>
     <install>
        <osTypes>
 <osType name="NTX86" startCmd="setup" timeout="600" successCodes="0,1,2"/>
        </osTypes>
    </install>
    <uninstall>
       <osTypes>
          <osType name="NTX86" startCmd="setup -u" timeout="30" successCodes="0"/>
        </osTypes>
   </uninstall>
</Deployment>
```

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"/>
```

2. Use the install section to configure the command to run after the package is uncompressed on a compute host.
3. For `startCmd`, 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:

```
startCmd="scripts\myscript"
```

4. Use the uninstall section to configure the command to run if the `startCmd` specified in the install section fails, or before the package is removed from a compute host.
5. Add `deployment.xml` to your service package with the executables for the commands you specified in `StartCmd`.

Important:

There can only be one deployment.xml file per service package. The file must be at the top level of the service package—it cannot be in a subdirectory.

6. Deploy the service package.
   a) In the Platform Management Console, select Manage Service Packages > Global Actions > Add Package to Repository.

The Add Package to respository page displays.

b) Browse to your service package and select it.
c) Select the application associated with your service package, then Add.

Your service package should now be displayed in the list.

Note:

You can also use the following commands:

soamdeploy add SampleService -p SampleService.exe.gz -c /SampleApplications/SOASamples

soamdeploy view -c /SampleApplications/SOASamples

Linux/UNIX service package

1. Create a file for your service package with the name `deployment.xml`.

   The file must be called `deployment.xml`.

   For example:

```
<Deployment xmlns="http://www.platform.com/Symphony/Deployment>
   <install>
     <osTypes>
     <osType name="LINUX86" startCmd="setup" timeout="600" successCodes="0,1,2"/>
      </osTypes>
   </install>
   <uninstall>
     <osTypes>
          <osType name="LINUX86" startCmd="setup -u" timeout="30" successCodes="0"/>
       </osTypes>
   </uninstall>
</Deployment>
```

Note:

All values in `deployment.xml` are case-sensitive when the service is deployed on Linux/UNIX.

---

2. Use the install section to configure the command to run after the package is uncompressed on a compute host.
3. For `startCmd`, 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:

```
startCmd="scripts/myscript"
```

4. Use the uninstall section to configure the command to run if the `startCmd` specified in the install section fails, or before the package is removed from a compute host.
5. Add `deployment.xml` to your service package with the executables for the command you specified in `StartCmd`.

Important:

There can only be one deployment.xml file per service package. The file must be at the top level of the service package—it cannot be in a subdirectory.

---

6. Deploy the service package.

a) In the Platform Management Console, select Manage Service Packages > Global Actions > Add Package to Repository.

   The Add Package to respository page displays.

b) Browse to your service package and select it.

c) Select the application associated with your service package, then Add.

   Your service package should now be displayed in the list.

Note:

You can also use the following commands:

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

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

Feature: Execution tasks integration

This feature 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.

Scope

Operating system	Linux, Windows, and Solaris hosts supported by Symphony
Limitations	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.

About the Symphony execution task feature

The Symphony execution task feature provides the ability to deploy existing executables in a grid environment, thereby realizing the benefits of Symphony without necessarily redeveloping your application code. If you have existing executables but are developing new clients, the Symphony SDK API can be used to start and control the remote execution of executable files. Each application consists of an execution service and an executable file that are distributed among compute hosts.

The execution service is common to all execution applications and is primarily responsible for starting the remote execution tasks and returning results to the client. When you install Symphony or Symphony DE, the execution application is pre-deployed but disabled.

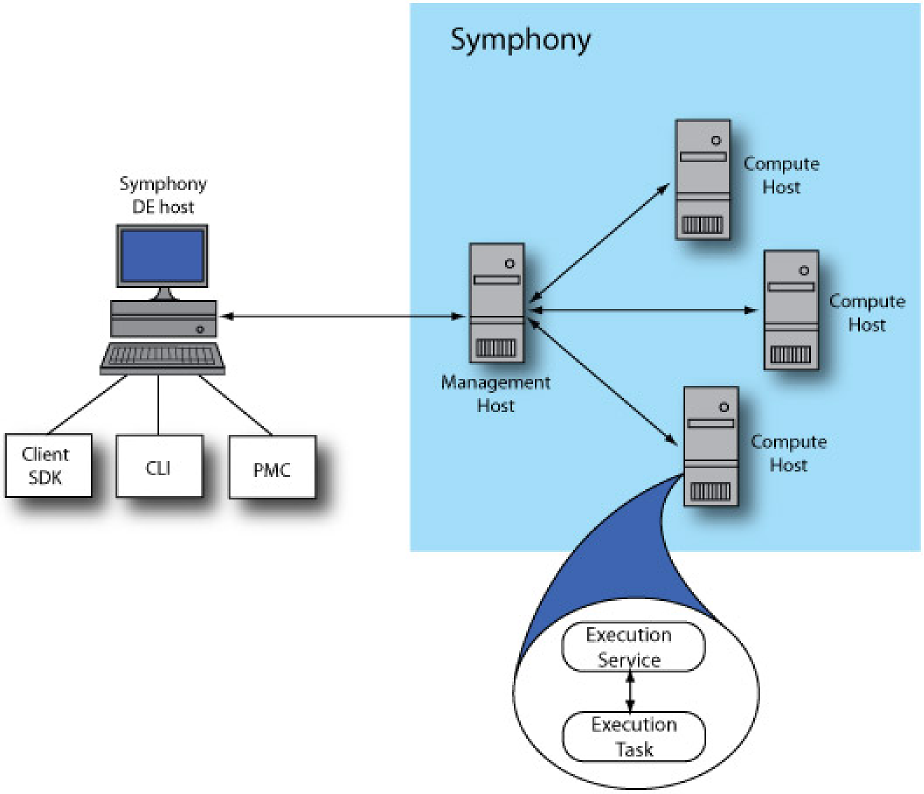
Here are the characteristics of a Symphony execution application:

- only exit codes from the execution tasks are returned to the client. Note that the execution service, which manages the execution tasks, can still return exceptions
- the execution task is started and stopped by the service process for every task, which has more overhead than the method invocation of the ordinary Symphony task within the service process.

There are three ways to run, control, and monitor execution tasks from the client host:

- Platform management console (PMC)
- command line interface (CLI)
- client SDK

Each interface offers a different level of functionality. For example, with the client SDK you can spawn a second thread to issue a non-blocking fetchTaskStatus() call while the client performs other functions. Or you can use the CLI for scripting execution task commands. For more information, refer to the Interfaces section of this chapter.



For the execution application to work, the executable files must reside on the same compute host as the execution service or at least be accessible from the compute host through a file sharing system or remote shell. Symphony's service deployment tools can be used to package the executable files and dependencies, and distribute them to the compute hosts.

To help you decide whether the execution application is right for your needs, here is the criteria you should use:

- you have an existing executable and want to reuse it without changing it to integrate with the Symphony API
- you want to write a script to launch the executable and run it on Symphony
- you have no access to the source code and cannot change it

Application execution flow

To help grasp the concepts, let's look at the functional flow of an execution application.

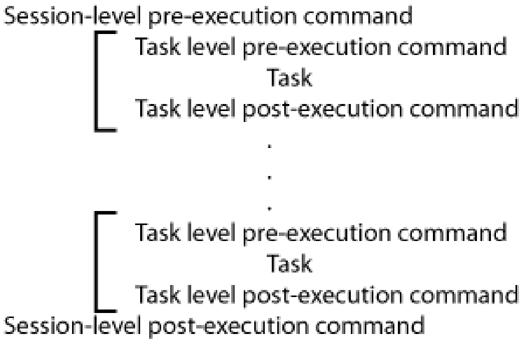
1. The client application creates a Symphony execution session and passes the following optional session-level data to the execution service:

    - array of environment variable strings in "name=value" pairs format (can be empty)
    - pre-execution command string (can be empty)
    - post-execution command string (can be empty)

    The execution service ensures that if the pre-execution command completes successfully, the post-execution command will always be executed, even if errors occur in the user's command during the session.

2. The client application sends the execution tasks to the execution service, which includes:
   - an executable command string with arguments
   - optional execution task context data including environment variable strings in "name=value" pairs format and pre-/post- execution commands

     Pre- and post-execution commands sequence:


Session-level pre-execution command
Task level pre-execution command
Task
Task level post-execution command
.
.
.
Task level pre-execution command
Task
Task level post-execution command
Session-level post-execution command

3. Upon receiving the input message on the service side, the execution service spawns a new process based on the execution task data.

   While the execution task process is running, the execution service periodically checks for interruptions and suspends or aborts the running process if it detects an administrative action such as session suspend or abort was issued by the CLI or PMC.
4. When the execution task has completed its process, the exit code of the process is sent back to the client in the execution task status. The error handling associated with this exit code is configurable in the application profile.

   Note:

   The exit code of pre- and post-commands is only logged, and not sent back to the client.

Execution service

The execution service implements the process execution logic as a Symphony service and interacts directly with the service instance manager (SIM).

To allow the execution task to use some unique identities, the service side execution environment is supplemented with the following additional environment variables:

- SYM_SERVICE_NAME = execution service name
- SYM_SERVICE_PID = execution service instance OS process identifier
- SYM_SESSION_ID = session identification number
- SYM_TASK_ID = task identification number

The execution service logs session and command start-up and shutdown audit information in a log file. The path for log files is configurable in the application profile. By default, the log is set to the INFO level, which includes the following data:

- date/time stamp
- local time zone
- host name
- service instance process identifier

- session identifier (for session pre- and post- commands)
- execution task identifier (for command-level executions)
- execution level (session or task)
- execution context (pre-, post-, or command execution)

Interfaces

This section describes the three interfaces that are available for sending execution tasks.

Client SDK

The Symphony execution task feature extends the existing Symphony API classes for C++ and Java languages:

Classes:

- ExecutionSession: a class to enable the client to manage its execution workload
- ExecutionEnumItems: container class for ExecutionStatus objects
- ExecutionStatus: container class for the status of an execution task result
- ExecutionSessionContext: container class for environment variables, session type, and pre-/post- commands for session-level context
- ExecutionCommandContext: container class for environment variables and pre-/post- commands for task-level context

Command line interface

Use symexec for the execution application operations. The symexec utility supports the following commands:

- create - creates an execution session in which to run an execution task. The session stays open until it is explicitly closed (detachable session)
- send - sends an execution task command in the execution session
- fetch - fetches the statuses of finished execution tasks in the execution session
- close - closes the execution session
- run - creates an un-detachable execution session, runs an execution task, waits and then retrieves the result, and closes the execution session.

For more details such as command syntax and options, refer to the Command Reference.

Platform management console

The management console implements a GUI panel that allows a single execution task to be sent per session. The interface also supports the entry of associated pre- and post- commands, as well as environment variables and their values. Basically, the functionality offered by this panel is identical to the `run` subcommand of the symexec CLI where "`Guest`" is the username and password. If you need to send multiple concurrent execution tasks, additional GUI dialogs must be opened.

The management console can retrieve log files that are created by the execution service for each execution session.

For more details, refer to the online help provided with the management console.

Configuration

Application profile

Every application requires an application profile to dynamically configure the application’s behavior within Symphony. An application profile for the default symexec application is already registered (but disabled) when Symphony is installed. There is also a second application profile, specific to execution applications, included in the Symexec sample that is packaged with Symphony.

The execution service is common to all execution applications. In cases where different consumers are running execution applications, the name of the application specified in the application profile must be unique. Also, each application profile name must be unique in the cluster and associated with only one consumer. At any time, there can be only one enabled application per consumer.

For the execution application to work properly, the following key parameters must be configured in the application profile:

- recoverable

  Set this flag to `true` since the execution application must be recoverable to support suspend/resume operations.
- suspendGracePeriod and taskCleanupPeriod

  Set both parameters so that the value is greater than the time it takes to complete the post-execution command that is executed by the service’s `onSessionLeave()` method. This time period allows for post-execution cleanup.
- controlCode

  In the application profile, the SessionEnter and SessionLeave Method elements correspond to the service methods `onSessionEnter()` and `onSessionLeave()`. It is within these methods that the session-level pre- and post- execution commands are executed. Similarly, the Invoke Method element corresponds to the `onInvoke()` service method where task-level pre- and post- execution commands, and the execution task’s command itself are executed. For the default execution application, its application profile is configured so that if the command's exit code has a value of zero, it will be treated as successful completion and any other value will be treated as a failure of the corresponding execution task.

  When an execution task’s command or session-level pre- and post- command is able to start and finish, the execution service stores the command’s exit code as a Symphony service context’s control code and then adds 1 to the control code. Therefore you may configure the control code for the Return event in the application profile with a value of 1 greater than the exit code you are expecting from the given command. For example, the default execution application is configured that when the command executes successfully with an exit code of 0, it sets the control code for this result to 1 (exit code +1) so that Symphony processes the result as a successful action.

  Note:

  The exit codes for task-level pre- and post- execution commands are not stored so consequently the handling of these exit codes cannot be configured.

  The default value for all unspecified control codes is 0. It means that if you have specified control code equal to 1 to indicate successful action, all failures or system errors that do not have a corresponding control code in the application profile will be automatically associated with control code 0.

  If the command cannot be started by the operating system, the execution service sets the control code to the system error number returned by the operating system. You can configure the Failure events in the application profile with control codes to handle specific exceptions. In this case, if the matching control code is configured as a failure, the exception message with the error code will be propagated back to the client. If an exception occurs with a control code that is not configured, it will automatically be associated with the action for control code 0.

> If your execution application has more than one possible exit code to indicate successful completion, you must configure these multiple control codes in the application profile associated with the successful action.

If you are using multiple application profiles for multiple execution applications, you can configure the logDirectory parameter in each profile so that each application stores logs in a unique location. Another way of organizing log files per application is to use different service names and update the respective application profile with the new service name. All application profiles use the service name in the sub directory path (subDirectoryPattern) for logging and this path is fixed on the service side for compute hosts. Changing the service name for each application profile is another way to ensure a unique location for each application's log files.

Logging

Logging that is configurable by level and class name can be enabled for the execution service itself using a separate section in the `api.log4j.properties` file. It is located in `$SOAM_HOME/conf` (Linux) or `%SOAM_HOME%\conf` (Windows) on the host where the execution service runs.

Run multiple services in an application

Goal

In your application, different sessions may need different services to perform computations.

You want to be able to specify that an application can run several services.

Assumptions

For the procedures in this document, it is assumed you want your application to use two different services, ServiceA, and ServiceB. The application is registered under the consumer `/SampleApplications/SOASamples`.

Package and deploy your services

Different services can use separate service packages or the same service package. Ensure each service definition in an application profile has a unique service name. Note that only one of your services can be set as the default service.

This example assumes you will create a separate package for each service used by your application.

For example, to use ServiceA and ServiceB in your application:

- Create `ServiceApkg.gz` and include all related binaries for ServiceA in this package.
- Create `ServiceBpkg.gz` and include all related binaries for ServiceB in this package.

1. Go to the directory that contains your service binaries and compress the service binaries into two files: `ServiceApkg.gz`, and `ServiceBpkg.gz`.
2. Deploy the service packages in the consumer with the `soamdeploy` command. (If you prefer, you may use the Wizard for this task.)

```
soamdeploy add ServiceApkg -p ServiceApkg.gz -c /SampleApplications/SOASamples
```

```
soamdeploy add ServiceBpkg -p ServiceBpkg.gz -c /SampleApplications/SOASamples
```

   The service packages are deployed.
3. Check the list of deployed services with the `soamdeploy view` command.

   For example:

   soamdeploy view -c /SampleApplications/SOASamples

   You should be able to see your service packages deployed. Notice that the Application field has a dash (-), indicating that there are no applications associated with the package you deployed.

Associate your application with the service packages

To associate your application with the different service packages, edit your application profile.

Important:

By configuring more than one service in your application, a host blocked because of an error in one of the services is also blocked for all other services of the same application.

1. Open your application profile.
2. In the session type definition, under `SessionTypes`, specify the session type name and add the `serviceName` parameter.

The `serviceName` parameter can be any name you want. It is used to link the session type definition with the service definition. If `serviceName` is not defined, the session uses the default service.

For example:

```
...
<SessionTypes>
        <Type name="MysessiontypeA" serviceName="ServiceA" priority="1"
recoverable="false" sessionRetryLimit="3" taskRetryLimit="3"
abortSessionIfTaskFail="false" suspendGracePeriod="100"
taskCleanupPeriod="100"persistSessionHistory="all" persistTaskHistory="all"/>
       <Type name="MysessiontypeB" priority="1" recoverable="false"
serviceName="ServiceB" sessionRetryLimit="3" taskRetryLimit="3"
abortSessionIfTaskFail="false" suspendGracePeriod="100"  taskCleanupPeriod="100"
persistSessionHistory="all" persistTaskHistory="all"/>
</SessionTypes>
```

3. In the service definition, under Service, specify the service name, service package name, and start command for the service. All services specified in SessionTypes should be configured in the Service section.

   a) For Service name, specify the same value that you specified for `serviceName` in the session type.

   b) Specify the packageName parameter and specify the name of the package you deployed.

      You can find out the package name with the command `soamdeploy view`.

   a) Change startCmd to point to your service executable.

      Leave the ${SOAM_DEPLOY_DIR} in your path as this is the deployment directory in the system. If your service is located under a subdirectory, indicate the subdirectory after ${SOAM_DEPLOY_DIR} in the path.

      On Windows:

```
<Service name="ServiceA" description="My Sample Service A"
packageName="ServiceApkg" deploymentTimeout="300">
        <osTypes>
            <osType name="all" startCmd="${SOAM_DEPLOY_DIR}\ServiceA.exe">
            </osType>
        </osTypes>
    </Service>
```

      On Linux:

```
<Service name="ServiceB" description="My Sample Service B"
packageName="ServiceBpkg" deploymentTimeout="300">
        <osTypes>
            <osType name="all" startCmd="${SOAM_DEPLOY_DIR}/ServiceB">
            </osType>
        </osTypes>
    </Service>
```

4. Repeat steps 2-3 for every service that you want to refer to in your application.

5. Add a `default` attribute for the service that you want to designate as the default so that it is started when the service instance manager starts.

   When specifying multiple services, you must designate one service as the default.

   For example, for ServiceA and ServiceB, your application profile should look similar to the following. Note that in this example, ServiceA is the default service.

```
<?xml version="1.0" encoding="UTF-8"?>
<Profile xmlns="http://www.platform.com/Symphony/Profile/Application" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" name="">
<Consumer applicationName="SampleApplicationCPP"
consumerId="/SampleApplications/SOASamples" policy="R_Proportion"
taskHighWaterMark="1.0" taskLowWaterMark="1.0" resourceBalanceInterval="5"
sessionSchedulingInterval="500" resourceGroupName="ComputeHosts"/>
...
```

```
<SessionTypes>
        <Type name="MysessiontypeA" priority="1" recoverable="false"
serviceName="ServiceA" sessionRetryLimit="3" taskRetryLimit="3"
abortSessionIfTaskFail="false" suspendGracePeriod="100"  taskCleanupPeriod="100"
persistSessionHistory="all" persistTaskHistory="all"/>
        <Type name="MysessiontypeB" priority="1" recoverable="false"
serviceName="ServiceB" sessionRetryLimit="3" taskRetryLimit="3"
abortSessionIfTaskFail="false" suspendGracePeriod="100"  taskCleanupPeriod="100"
persistSessionHistory="all" persistTaskHistory="all"/>
</SessionTypes>

<Service name="ServiceA" description="The Sample Service A"
packageName="ServiceApkg" default="true" deploymentTimeout="300">
        <osTypes>
            <osType name="NTX86" startCmd="${SOAM_DEPLOY_DIR}\ServiceA.exe">
        </osType>
        </osTypes>
    </Service>

<Service name="ServiceB" description="The Sample Service B"
packageName="ServiceBpkg" deploymentTimeout="300">
            <osTypes>
                <osType name="LINUX86"startCmd="${SOAM_DEPLOY_DIR}/ServiceB">
                </osType>
            </osTypes>
        </Service>
</Profile>
```

6. Register the application profile with the `soamreg` command. (If you prefer, you may use the Wizard for this task.)

   For example:

   ```
   soamreg SampleApp.xml
   ```

   The application is registered and enabled.
7. Check that the application is associated with the package with the `soamdeploy view` command.

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

   You should be able to see package names and the associated application names.

Check your client application code and run your client

Check your client application code to ensure:

- The application name specified when connecting to the application is the same as that specified in the application profile
- The session types you specified to create the session must exist in your application profile unless you specified " " for the session types, which means to use the default session type.

  If you want to specify the service name directly to override the service configured in the session type, the service must be configured in your application profile.

1. Check client code to ensure the application name specified in `connect( )` is the same as that specified in the application profile.

   For example, if, in your application profile you have `applicationName="SampleAppCPP"`,

   your client code must also contain `SampleAppCPP`:

```
...
// set up application specific information to be supplied to the System
ConnectionPtr conPtr = SoamFactory::connect("SampleAppCPP",&securityCB);
...
```

2. Check client code to ensure the session type name specified when creating the session is configured in your application profile.

   For example, create sessions with the session types for ServiceA and ServiceB:

```
...
// Set up session creation attributes for ServiceA
SessionCreationAttributes attributesA;
attributesA.setSessionName("mySessionA");
attributesA.setSessionType("MysessiontypeA");
attributesA.setSessionFlags(SF_RECEIVE_SYNC);
// Set up session creation attributes for ServiceB
SessionCreationAttributes attributesB;
attributesB.setSessionName("mySessionB");
attributesB.setSessionType("MysessiontypeB");
attributesB.setSessionFlags(SF_RECEIVE_SYNC);
// Create synchronous sessions
SessionPtr sesPtrA = conPtr->createSession(attributesA);
SessionPtr sesPtrB = conPtr->createSession(attributesB);
...
```

3. Save your client code and recompile.
4. Run your client to submit work to your application.

Remove an application

You want to remove application binaries from the system.

When you remove an application through the Platform Management Console, the application is unregistered, the associated service package removed from the repository, and in Symphony grid, the associated consumer deleted.

Unregistering the application:

- Terminates existing sessions and tasks
- Releases all resources allocated to the application
- Unregisters the application
- Removes the service package from the repository if it is not shared with any other application(s)

Note:

You can also use the `soamunreg` command to unregister an application, and `soamdeploy remove` to remove the service package. For example:

```
soamunreg SampleAppCPP
```

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

1. In the Platform Management Console, select Symphony Workload > Configure Applications.

   The Applications page displays.
2. Select Global Actions > Add/Remove Applications.

   The Add/Remove Application page displays.
3. Select Remove an existing application, then Continue.

   The Remove an application page displays.
4. Follow the prompts.

General file package deployment

If you have files, such as patches, libraries, or data that you need to deploy to several hosts in the cluster, you can use the repository server to simplify the process. You do this by uploading the installation files to the repository server, then using the `rsdeploy` command to launch the installation to a group of hosts.

Limitations of rsdeploy

The repository server and the rsdeploy agents do not support the following:

- Failover and recovery of in-progress application deployment
- Deployment to shared-file system EGO installations
- Package version control

  You need to provide a different package name for each version of the file package.
- File package validation
- GUI integration

  The rsdeploy command can only be launched from the command line.

File package deployment scripts

The `rsdeploy` agent on each host launches scripts to install or uninstall a file package on each host. These scripts are defined in the `deployment.xml` configuration file, and must be included in the file package.

The install and uninstall scripts can use the environment variables defined in the `deployment.xml` file. They can also use the following variables:

Linux/UNIX environments:

- EGO_TOP
- $SOAM_HOME
- $EGO_BINDIR
- $EGO_LIBDIR
- $EGO_SERVERDIR
- $EGO_LOCAL_CONFDIR

Windows environments:

- EGO_TOP
- %SOAM_HOME%
- %EGO_BINDIR%
- %EGO_LIBDIR%
- %EGO_SERVERDIR%
- %EGO_LOCAL_CONFDIR%

The scripts should assume that all the extracted files will match the file paths in the file package (under the `staging` directory before being compressed). Using the script, the files can be copied to any location on the local hard disk or any location defined by a variable. The execution user for the script is the user assigned to the consumer specified in the `rsdeploy install` or `rsdeploy uninstall` command.

To save disk space, the install script should remove any unnecessary files from the deploy location; however, it must not remove the `deployment.xml` and uninstall scripts, as these files are required to uninstall the file package.

File package deployment configuration file

When installing a file package, the `rsdeploy` agent on each host retrieves the `deployment.xml` configuration file from within the file package and executes the specified installation command. Likewise, when uninstalling a file package, the `rsdeploy` agent executes the specified uninstallation command.

The following is an example of the `deployment.xml` file, which defines installations for Windows and Linux environments running on x86 machines (OS types NTX86 and LINUX86):

```
<?xml version="1.0" encoding="UTF-8"?>
<Deployment xmlns="http://www.platform.com/Symphony/Deployment>
 <install>
  <osTypes>
    <osType name="NTX86" startCmd="cmd /c cmd /c wininst.bat" timeout="600" successCodes="0,1,2,3">
        <env name="WIN_ENV">c:\temp</env>
    </osType>
    <osType name="LINUX86" startCmd="/bin/sh linuxinst.sh" timeout="600" successCodes="0,1,2,3">
        <env name="LINUX_ENV">/tmp</env>
    </osType>
  </osTypes>
 </install>
 <uninstall>
  <osTypes>
    <osType name="NTX86" startCmd=" cmd /c cmd /c winuninst.bat " timeout="600"
successCodes="0,1,2,3">
        <env name="WIN_ENV">c:\temp</env>
    </osType>
    <osType name="LINUX86" startCmd="/bin/sh linuxuninst.sh" timeout="600" successCodes="0,1,2,3">
        <env name="LINUX_ENV">/tmp</env>
    </osType>
  </osTypes>
 </uninstall>
</Deployment>
```

When creating a `deployment.xml` file, consider the following:

- The `deployment.xml` file must be in the root location in the file package.
- The startCmd current working directory is set to the deployment directory.
- Success codes are compared to the exit code of the startCmd. If the successCodes attribute is not defined, there is no exit code checking.
- The `<env>` section can be used to set up the environment for the install and uninstall commands.
- In a mixed cluster, each package can be associated with a specific OS type (using the `-o` option with `rsdeploy`). If the file package is to be installed on multiple OS types, install and uninstall sections should include all the supported OS types.
- The exit code of a script is usually the exit code of the last command that the script executes.
- The timeout periods for the install and uninstall scripts are specified in seconds. The default is no timeout period.

Directories

Packages

The repository server caches packages in the *`EGO_TOP`*`/eservice/rs/.cache/.global` directory. The directory can be modified by changing RS_STORAGE_DIR in the `rs.xml` service file. If your cluster uses a shared EGO directory, the repository server cache directory is located under the EGO shared directory.

The repository server deploys packages in the *`EGO_TOP`*`/eservice/rs/deploy/.global` directory on each host.

Logs

The repository server logs to the *`EGO_TOP`*`/eservice/rs/log` directory. The debug level can be configured by modifying the *`EGO_TOP`*`/eservice/rs/conf/rs.log4j.properties` file. For example, change INFO to DEBUG in the following line, by changing:

```
"log4j.rootLogger=INFO, RS"
```

to

```
"log4j.rootLogger=DEBUG, RS"
```

rsdeploy agent

The rsdeploy agent starts from the *`EGO_TOP`*`/eservice/rs/work` directory, and logs to that location. The rsdeploy agent starts the install/uninstall command from the deployed location directory.

File deployment logging

File deployment uses the RS EGO service running on the master host, and an rsdeploy agent that runs on each compute host.

The RS EGO service generates an `rs.`*`hostname`*`.log` file under `/eservices/rs/log` on the host on which the RS service runs. You can use the Platform Management Console to retrieve this Symphony log file.

The rsdeploy agent generates a `cli.log` file in the `/eservice/rs/work` directory on the compute host. You can use the Platform Management Console generic log retrieval feature to retrieve this log file.

The following is an example of a log file identifying a problem:

```
RSAuditlogger - 2007-10-02 17:56:45.325 Eastern Daylight Time CONFIG <Admin> PACKAGE
<test> HOST <-> install failed - Domain <Application>: The specified resource
ComputeHost has no hosts or was not found in the cluster. Ensure that the resource
group exists in EGO and contains hosts.
```

Enable logging for the rsdeploy agent

1. Copy a log4j properties file called `cli.log4j.properties` to `/eservice/rs/work` on the target host.

This file shows errors relating to decompressing the file package and running the install script.

The following example is the resulting log when the `winuninst.bat` file was missing from the package:

```
CLI.dpl.download.RunTask - Domain <Application>: The command,  cmd /c cmd /c
winuninst.bat , failed with the error, The system cannot find the file
specifiedCLI.dpl.download.InstallModule - Domain <Application>: Failed to
start the uninstall command,  cmd /c cmd /c winuninst.bat .
```

Deploy a file package using the repository server

Use rsdeploy to deploy a file package to several hosts in the cluster.

The following steps summarize the process:

1. Create a file package containing the files to deploy, a deployment configuration file called `deployment.xml`, an install script, and an uninstall script.
2. Upload the file package to the repository server by sending an add request to the repository server.

3. Deploy the file package to a group of hosts by sending an install request to the repository server.

   The repository server installs the file package by remotely starting the `rsdeploy` agent on each host. The `rsdeploy` agent downloads the package and executes the install script to install the files onto the host.
4. If you want to remove the file package from the group of hosts, send an uninstall request to the repository server.

   The repository server remotely removes the file package by remotely starting the `rsdeploy` agent on each host. The `rsdeploy` agent runs the uninstall script to remove the files from the host, then removes the file package.
5. If you no longer need the file package, remove it from the repository server.

Create a file package

Create a file package to deploy files to multiple hosts in your cluster.

1. Plan how the files will be installed on each target host.
   a) Determine which files should be in the file package.
   b) Determine where the files go on each target host.
   c) Decide what install commands need to be run on each target host.
2. Create a temporary staging directory to contain the file package.

   For example, create a directory called `staging`.
3. Copy all required files to the staging directory, placing them in subdirectories to match their final location on the target host.

   For example, if you are assembling the application package on Linux/UNIX, and Windows hosts,

   - `/staging/root/files1/*`
   - `/staging/root/files2/*`
   - `/staging/deployment.xml`
   - `/staging/linuxinst.sh`
   - `/staging/linuxuninst.sh`
   - `/staging/solinst.sh`
   - `/staging/soluninst.sh`
   - `/staging/wininst.bat`
   - `/staging/winuninst.bat`
4. Compress all the files in the staging directory into a file package.

   When compressing the files, consider the following:

   - Use relative paths and do not include the staging directory itself in the package.
   - `rsdeploy` supports `.tar.Z`, `.tar.gz`, `.tar.zip`, `.tgz`, `.taz`, `.tar`, `.jar`, `.gz`, and `.zip` files by starting a command-line utility to extract the file package. However, if the file package will be deployed on Windows target hosts, avoid using the tar format unless you installed a tar utility on all Windows hosts.

Upload a file package to the repository server

Add a file package to the repository server for installation to hosts in the cluster.

On each target host, the `rsdeploy` agent downloads this file package from the repository server prior to installation.

1. Use the `rsdeploy add` command to add the package to the repository server.

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

   Because `rsdeploy` does not directly support package versioning, append the package version to the package name. Later versions of the package should either include all of the files from the previous version, or include commands in the install script to check whether the required packages are installed (and fail the installation if they are not).

   For example,

   - To deploy a file package intended for all hosts in the cluster:

     rsdeploy add myapp.v1 -p myfile_package.zip
   - To deploy a file package intended for all `LINUX86` hosts in the cluster:

     rsdeploy add myapp.v1 -p myfile_package.tar.gz -o LINUX86

Deploy a file package from the repository server

Deploy a file package from the repository server to hosts in the cluster.

1. Use the `rsdeploy install` command to install a file package to multiple hosts in the cluster.

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

   For example, to install the file package with the default settings:

   rsdeploy install mypkg.v1

   The repository server attempts to install `mypkg.v1` (using the install script included with the file package) on all hosts in the cluster using the default consumer `/ClusterServices/EGOClusterServices` and the resource group `InternalResourceGroup`.

   The execution user for the installation is the user associated with the consumer, normally the installation user.
2. To view the status of the install, use the `rsdeploy status` command.

   rsdeploy status *package_name* [**-s** *status_filter*] [**-u** *user_name*] [**-x** *password*]

   where *status_filter* is the status of the file package that you want to view. Valid values are `all`, `allocating`, `waiting`, `active`, `done`, `error`, and `cancelled`. If `-s` is unspecified, this filter defaults to `all`.

   For example,

   - To view the status of the file package installation on all hosts:

     rsdeploy status mypkg.v1
   - To view the hosts with canceled file package installation:

     rsdeploy status mypkg.v1 -s cancelled

If some hosts were unavailable during the installation, run `rsdeploy install` again when the hosts become available. The repository server only attempts to install the file packages on hosts that do not have the package installed.

Note:

The repository server does not automatically install the file package to unavailable hosts when they become available. You need to manually run `rsdeploy install` when the hosts become available.

Uninstall a file package from hosts

Uninstall a file package that you installed using the repository server from multiple hosts.

1. Use the rsdeploy uninstall command to uninstall a file package.

   rsdeploy uninstall *package_name* [**-c** *consumer* **-r** *resource_group* | **-r** *resource_group*] [**-t** *host_name*] [**-u** *user_name*] [**-x** *password*]

   For example, to uninstall the file package with the default settings,

   rsdeploy uninstall mypkg.v1

   The repository server attempts to uninstall `mypkg.v1` (using the uninstall script included with the file package) on all hosts in the cluster using the default consumer `/ClusterServices/EGOClusterServices` and the resource group `InternalResourceGroup`.

   The execution user for the installation is the user associated with the consumer, normally the installation user. The execution user must have sufficient access rights to run the uninstall script and remove the directory.

Uninstalling the file package does not remove it from the repository server.

If some hosts were unavailable during the uninstallation, run `rsdeploy uninstall` again when the hosts become available or use `rsdeploy status` to view the status of the uninstall.

Note:

The repository server does not automatically uninstall the file package from unavailable hosts when they become available. You need to manually run `rsdeploy uninstall` when the hosts become available.

Remove a file package from the repository server

1. Use the `rsdeploy remove` command to remove a file package from the repository server.

   rsdeploy remove *package_name* [**-c** *consumer*] [**-o** *os_type*]

   For example, to remove a file package from the repository server:

   rsdeploy remove mypkg.v1

   To remove only the linux86 file package from the repository server:

   rsdeploy remove mypkg.v1 -o LINUX86

Troubleshooting file package deployment issues

The repository server uses the specified installation consumer to obtain resources and spawn rsdeploy agents on remote hosts. The rsdeploy agent signals back any errors or exception conditions to the repository server.

If the repository server cannot obtain a slot on the target host, or if the `rsdeploy` command fails to start on the specified host, the status is a timeout error

- To see information about the error, use `egosh activity list` or `egosh activity view` and look for the rsdeploy agent command for the specific host.
- To debug the install or uninstall script, manually start the script in a testing location to ensure that it has the expected behavior. Put `echo "debug">`*`/tmp/file`* statements into the script to aid in debugging.
- To ensure that all the files are copied correctly to the compute host, remove the `deployment.xml` file from the package, and install the package on a single host. This eliminates problems with the install and uninstall scripts. If the install script fails to run, or returns an error code, the files in the `/deploy` directory are removed.
- The version of unzip shipped with EGO may not be able to decompress all long file names under Windows. Check whether EGO's `/bin/unzip` command can decompress the package if it has long file names.

Updating applications

This section describes how to update a Symphony application. There are two approaches to updating applications: static configuration and dynamic configuration. The one you choose depends on the scope of your changes.

Static configuration updates allow you to modify any parameter within the application profile. It offers a wider range of possible changes to application profile parameters than the dynamic configuration update but requires that the application be reregistered after the changes. This results in the termination of running workload associated with the application.

Dynamic configuration updates allow you to update an application without impacting existing clients or workload. Using this method, only changes to the service and session type sections of an application profile can be made. For other changes, the application must be disabled and unregistered, which results in the termination of running workload for that application.

Static configuration update

Use this method of updating an application when you want to change some application parameters and the changes are beyond the scope of a dynamic configuration update.

Updating the application profile does the following:

- Terminates existing sessions and tasks
- In Symphony grid, releases all resources allocated to the application and shuts down session manager. In Symphony DE, there is no resource allocation.
- Re-registers the application profile

Note:

You can also update application parameters through the command-line. Use `soamview app appname -p >`*filename* to send your current profile to a file. Edit it and use `soamreg` to re-register the application.

1. In the Platform Management Console, select Symphony Workload > Configure Applications.
2. Click the application name.

   The application profile displays.
3. From the drop-down list, select Basic Configuration or Advanced Configuration.
4. Modify desired parameters.

   Note:

   If parameters you want to modify are not visible through the Console, export the application profile and modify it with an XML editor, then reimport the updated profile.
5. Click Save to apply your changes and restart the application.

About dynamic application updates

This section provides an overview of the dynamic application update feature.

Symphony's dynamic application update feature facilitates the administration of service packages and their associations with applications. The Symphony application update features allow you to:

- Deploy an updated service package without stopping the current workload.
- Dynamically update or remove session type and service sections from the application profile without stopping the current workload. Only sessions using the removed sections are affected.
- Dynamically add session type and service sections to the application profile without stopping the current workload.
- Deploy a service package to any level of the consumer tree allowing the service to be shared with all consumers below this level. This enables service packages to be shared among multiple applications linked by the downward path of the consumer hierarchy.

The following table offers guidelines for choosing the right method to perform typical application updates. Within this table, the term workload is defined as existing running tasks and sessions associated with the application that is being updated.

Option	What you want to do	Result
1	**1.** You have an updated service package for an enabled application. **2.** You want current and future workload to use the updated service package. **3.** You want to overwrite the original service package. **Note:** You will not be able to switch back to the original service package. Refer to *Using the PMC to deploy service packages for consumers with short names* on page 403	- Workload continues to run with the next scheduled task using the updated service package. - The updated service package has the same name as the original service package. - Once updated, the original service package is no longer available in the repository. - Clients do not need modification.
2	**1.** You have an updated service package for an enabled application. **2.** You want current and future workload to use the updated service package. **3.** You want to be able to easily switch back to the original service package, if necessary. **Note:** You must use a new name for the updated service package so that the repository can store both the original and updated packages. Refer to *Change a service package for an existing service* on page 404	- Workload continues to run with the next scheduled task using the updated service package. - The updated service package has a different name than the original serivce package. - Once it is replaced, the original service package is still available in the repository. - Clients do not need modification.
3	**1.** You have a new service package for an enabled application. **2.** You only want clients that have been notified to use the new service. Refer to *Add a new service and session type* on page 405	- Workload continues to run using the existing service. - Clients that are aware of the new session type can use the new service. - Clients may need modification.

Option	What you want to do	Result
4	**1.** You have a new service package for an enabled application. **2.** You want to use the new service when you create a new session. Refer to *Assign a new service to an existing session type* on page 408	• Workload continues to run using the existing service. • New sessions with the updated session type use the new service. • Clients do not need modification.
5	You no longer need a service or session type and want to remove it from an application. Refer to *Remove a service/session type* on page 410	Any session that uses the removed service or session type is aborted.

Using the PMC to deploy service packages for consumers with short names

.

1. From the PMC, select Symphony Workload > Manage Service Packages.
2. Select the relevant consumer (applicable to Symphony grid version only).
3. Select Global Actions > Add package to repository.
4. Click in the deploy use short consumer name checkbox.

Update an existing service package using the CLI

Perform this task when you want to overwrite an existing service package in the repository. For example, you made a modification to a service binary and would like to replace the existing serivce binary with the new one without disrupting existing clients or workload. In this case, when you redeploy a service package that is being used by an enabled application, workload continues to run with the next scheduled task using the updated service package.

1. Compile your new service binaries and add them to the service package.
2. At the command prompt, change your current directory to the directory where the service package is located.

   Note:

   For the following step, ensure that the package name matches the original package name in the repository.

3. Deploy the service package:

   For example:

   soamdeploy add SampleService -p SampleService.zip -c /SampleApplications/SOASamples
4. Verify that the workload is still running (applicable to long-running tasks):

   For example:

   soamview session SampleApp

Change a service package for an existing service

Perform this task when you want to replace a service package but you also want to be able to easily switch back to the original service package. Workload continues to run with the next scheduled task using the updated service package.

1. Compile your new service binaries and add them to the service package.
2. Associate the new service package with a service:
    a) From the PMC, select Symphony Workload > Configure Applications.
    b) Select the relevant consumer (applicable to Symphony grid version only).
    c) Click the application name.

       The application profile displays.
    d) From the drop-down list, select Dynamic Configuration Update.

       A sub menu displays.
    e) Select Change Service Package/Attributes.
    f) From the Service Package drop-down list, select the new service package.

       If the new service package is not in the list:

        1. Select Add Package to repository.
        2. Click Browse and navigate to the new service package. Select the package and click Open.
        3. Choose whether to use the file name as the package name or enter a new name. In either case, ensure that the package name is *different* than the package name you are replacing.
        4. Click Add.

           An information dialog displays. Click Close.

           The new service package displays in the drop-down list.
    g) Update the Start Command, if necessary.
    h) Click Apply.

       A confirmation dialog displays. Click OK.

       An information dialog displays. Click OK.
    i) Click Close.
3. Verify that the workload is still running (applicable to long-running tasks):
    a) Select Symphony Workload > Monitor Workload.
    b) Click the application name.
    c) Click the session ID.
    d) Verify that the tasks are still running. The update takes effect with the start of the next scheduled task.

Note:

If you need to switch back to the original service package, simply associate the service with the original service package, as described above.

Change a service package using the CLI

Perform this task when you want to replace a service package but you also want to be able to easily switch back to the original service package. Workload continues to run with the next scheduled task using the updated service package.

1. Compile your new service binaries and add them to the service package.
2. At the command prompt, change your current directory to the directory where the service package is located.

   Note:

   For the following step, ensure that the package name is *different* than the original package name in the repository.

3. Deploy the service package:

   For example:

   soamdeploy add SampleApp_pkg2 -p SampleApp_pkg2.zip -c /SampleApplications/SOASamples

4. Associate the new service package with the service:
   a) Open the application profile with an editor.
   b) Update the service package name. For example:

```
<Profile ...>
    <Service name="ServiceA" description="My Sample Service A"
  packageName="SampleApp_pkg2" deploymentTimeout="300">
    </Service>
...
</Profile>
```

   c) Save the file.

5. Register the application dynamically:

   For example:

   soamreg SampleApp.xml -d

6. Verify that the workload is still running (applicable to long-running tasks):

   For example:

   soamview session SampleApp

   Note:

   If you need to switch back to the original service package, simply associate the service with the original service package, as described above.

Add a new service and session type

Perform this task when you want to add a new service and session type to your application. For example, you want to restrict the use of a new service only to clients that have been notified about the new session type in your existing application. You want to add this new service and session type to the application without affecting existing clients or workload. New sessions created after this update can use the new service and session type.

This procedure assumes that you have already created a new service package.

1. Add a new service to the application:
   a) From the PMC, select Symphony Workload > Configure Applications.
   b) Select the relevant consumer (applicable to Symphony grid version only).
   c) Click the application name.

    The application profile displays.

  d) From the drop-down list, select Dynamic Configuration Update.

    A sub menu displays.

  e) Select Add Service/Session Type.
  f) In the Service Definition group, click Add.
  g) Enter the service name. Click Add.
  h) Enter a description for the service.
  i) From the Service Package drop-down list, select the new service package.

    If the new service package is not in the list:

    1. Select Add Package to repository.
    2. Click Browse and navigate to the new service package. Select the package and click Open. Click Add.

       An information dialog displays. Click Close.

  j) Update the start command.

    For example:

    ${SOAM_DEPLOY_DIR}/SampleService

2. Add a new session type to the application:
   a) In the Session Type Definition group, click Add.
   b) Enter the new session type. Click Add.
   c) From the Service Definition drop-down list, select the new service that you just added to the application.
   d) Click Apply.

      A confirmation dialog displays. Click OK.

      An information dialog displays. Click OK.

   e) Click Close.
3. Verify that the workload is still running (applicable to long-running tasks):
   a) Select Symphony Workload > Monitor Workload.
   b) Click the application name.
   c) Click the session ID.
   d) Verify that the tasks are still running. (Existing clients and workload are not affected.)

      You can use the new service and session type when you create a new session.

Add a new service and session type using the CLI

Perform this task when you want to add a new service and session type to your application. For example, you want to restrict the use of a new service only to clients that have been notified about the new session type in your existing application. You want to add this new service and session type to the application without affecting existing clients or workload. New sessions created after this update can use the new service and session type.

This procedure assumes that you have already created a new service package.

1. Add the new session type and service to the application:
   a) Open the application profile with an editor.

b) Add the new session type definition to the application by creating a new Type element.

c) Set the session type name and service name attributes.

The service name can be any name you want and is used to link the session type definition with the service definition.

For example:

```
...
<Profile ...>
  ...
    <SessionTypes>
      <Type name="MysessiontypeA" serviceName="SampleServiceA" priority="1"
      recoverable="false" sessionRetryLimit="3" taskRetryLimit="3"
      abortSessionIfTaskFail="false" suspendGracePeriod="100"
      taskCleanupPeriod="100"persistSessionHistory="all"
      persistTaskHistory="all"/>
    </SessionTypes>
```

d) Add the new service definition to the application by creating a new Service element.

e) Set the service name and package name attributes.

The service name must match the service name that you specified in the Type element.

f) Change startCmd to point to your service executable.

Leave the *${SOAM_DEPLOY_DIR}* in your path as this is the deployment directory in the system. If your service is located under a subdirectory, indicate the subdirectory after *${SOAM_DEPLOY_DIR}* in the path.

On Windows:

```
  <Service name="SampleServiceA" description="My Sample Service A"
 packageName="ServiceApkg" deploymentTimeout="300">
    <osTypes>
      <osType name="all" startCmd="${SOAM_DEPLOY_DIR}\SampleServiceA.exe">
      </osType>
    </osTypes>
  </Service>
...
</Profile>
```

On Linux:

```
  <Service name="SampleServiceA" description="My Sample Service A"
 packageName="ServiceApkg" deploymentTimeout="300">
    <osTypes>
      <osType name="all" startCmd="${SOAM_DEPLOY_DIR}/SampleServiceA">
      </osType>
    </osTypes>
  </Service>
...
</Profile>
```

g) Save the application profile.

2. Register the application profile dynamically with the `soamreg` command.

   For example:

   ```
   soamreg SampleApp.xml -d
   ```

   The application is updated, registered, and enabled. Existing clients and workload are not affected by the update. You can use the new service and session type when you create a new session.

3. Verify that the workload is still running (applicable to long-running tasks):

For example:

soamview session SampleApp

Assign a new service to an existing session type

Perform this task when you want to assign another service to an existing session type. For example, you have a new service and want to associate it with an existing session type. You want to use the new service starting with the next session.

Updating the session type in the application profile results in the following:

- Open sessions having the updated session type will continue to use the old service until the sessions are closed
- New sessions created after the update will use the new service

This procedure assumes that you have already created a new service package.

1. Add a new service to the application:
    a) From the PMC, select Symphony Workload > Configure Applications.
    b) Click the application name.

        The application profile displays.
    c) From the drop-down list, select Dynamic Configuration Update.

        A sub menu displays.
    d) Select Add Service/Session Type.
    e) In the Service Definition group, click Add.
    f) Enter the new service name. Click Add.
    g) From the Service Package drop-down list, select the new service package.

        If the new service package is not in the list:

        1. Select Add Package to repository.
        2. Click Browse and navigate to the new service package. Select the package and click Open. Click Add.
    h) Click Apply.
2. Associate the new service with an existing session type:
    a) Select Symphony Workload > Configure Applications.
    b) Click the application name.

        The application profile displays.
    c) From the drop-down list, select Dynamic Configuration Update.

        A sub menu displays.
    d) Select Change Service For Session Type.
    e) From the drop-down list in the Session Type Definition group, select the relevant session type.
    f) From the Service Definition drop-down list, select the new service you want to assign to the selected session type.
    g) Click Apply.

        A confirmation dialog displays. Click OK.

        An information dialog displays. Click OK.

   h) Click Close.
3. Verify that the workload is still running (applicable to long-running tasks):
   a) Select Symphony Workload > Monitor Workload.
   b) Click the application name.
   c) Click the session ID.
   d) Verify that the tasks are still running.

      The new service will take effect when you create a new session. Existing clients and workload will not be affected.

Assign a new service to an existing session type using the CLI

Perform this task when you want to assign another service to an existing session type. For example, you have a new service and want to associate it with an existing session type. You want to use the new service starting with the next session.

Updating the session type in the application profile results in the following:

- Open sessions having the updated session type will continue to use the old service until the sessions are closed
- New sessions created after the update will use the new service

This procedure assumes that you have already created a new service package.

1. Add the new service to the application and assign it to an existing session type:
   a) Open the application profile with an editor.
   b) Add the new service definition to the application by creating a new Service element.
   c) Set the service name and package name attributes.
   d) Change startCmd to point to your service executable.

      Leave the *${SOAM_DEPLOY_DIR}* in your path as this is the deployment directory in the system. If your service is located under a subdirectory, indicate the subdirectory after *${SOAM_DEPLOY_DIR}* in the path.

      On Windows:

```
    <Service name="ServiceA" description="My Sample Service A"
  packageName="ServiceApkg" deploymentTimeout="300">
        <osTypes>
            <osType name="all" startCmd="${SOAM_DEPLOY_DIR}\ServiceA.exe">
            </osType>
        </osTypes>
    </Service>
...
</Profile>
```

      On Linux:

```
    <Service name="ServiceA" description="My Sample Service A"
  packageName="ServiceApkg" deploymentTimeout="300">
        <osTypes>
            <osType name="all" startCmd="${SOAM_DEPLOY_DIR}/ServiceA">
            </osType>
        </osTypes>
    </Service>
...
</Profile>
```

e) Assign the new service to a session type by setting the serviceName attribute in the Type element to the service name that you specified in the Service element.

For example:

```
...
<Profile ...>
    ...
        <SessionTypes>
            <Type name="MysessiontypeA" serviceName="ServiceA" priority="1"
            recoverable="false" sessionRetryLimit="3" taskRetryLimit="3"
            abortSessionIfTaskFail="false" suspendGracePeriod="100"
            taskCleanupPeriod="100"persistSessionHistory="all"
            persistTaskHistory="all"/>
        </SessionTypes>
```

f) Save the application profile.

2. Register the application profile dynamically with the `soamreg` command.

   For example:

   ```
   soamreg SampleApp.xml -d
   ```

   The application is updated, registered, and enabled. Existing clients and workload are not affected by the update. You can use the new service when you create a new session.

3. Verify that the workload is still running (applicable to long-running tasks):

   For example:

   ```
   soamview session SampleApp
   ```

Remove a service/session type

Perform this task when you want to remove a service or session type and your application is already enabled.

All open sessions having the removed session type or service will be aborted.

1. Select Symphony Workload > Configure Applications.
2. Click the application name.

   The application profile displays.

3. From the drop-down list, select Dynamic Configuration Update.

   A sub menu displays.

4. Select Remove Service/Session Type.
5. From th relevant drop-down list, select the service name or session type you want to remove.
6. Click Remove.

   A confirmation dialog displays. Click OK.

7. Click Apply.

   A confirmation dialog displays. Click OK.

   An information dialog displays. Click OK.
8. Click Close.

   The service or session type, as applicable, is removed from the application profile.

Remove a service/session type using the CLI

Perform this task when you want to remove a service or session type and your application is already enabled.

All open sessions having the removed session type or service will be aborted.

1. Remove a service or session type from the application:
   a) Open the application profile with an editor.
   b) Delete the relevant session type or service definition.
   c) Save the application profile.
2. Register the application profile dynamically with the `soamreg` command.

   For example:

   ```
   soamreg SampleApp.xml -d
   ```

   The application is updated, registered, and enabled.

19

Managing Data

Scenario: Maintaining data affinity between a session and service instances

Goal

You have services that cache market data for calculations on compute hosts. Each service loads data into memory and this operation is time-consuming compared to the calculation. Once the data is loaded, it does not change, and it can be used for all calculations that are requested.

Use the minimum services (R_MinimumServices) 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.

Change your application profile for data affinity

With this scheduling policy, you define a minimum number of service instances to be allocated to a session, regardless of workload or priority of other sessions, and they continue to serve the session until the session is suspended, killed or closed.

Service instances additional to the minimum service instances are proportionally shared among sessions with pending tasks based on session priority. These service instances are allocated and reallocated to sessions based on priority. Sessions that do not have workload are not allocated additional service instances.

Note:

When configuring the R_MinimumServices policy with multiple session types for an application, each resource group name in the resource group filters should be unique among all of the session types; otherwise you may get one less than the configured number of minimum services running.

Note:

If you are editing the application profile outside the Platform Management Console, in the Consumer section, add the parameter `policy="R_MinimumServices"`. In the session types section, add the parameters `priority`, and `minServices` and register the application with the `soamreg` command.

1. In the Platform Management Console, click Symphony Workload > Configure Applications.

   The Applications page displays.
2. Select the application you want to modify.

   The Application profile page displays.
3. Select SSM scheduling policy to expand it, then under Policy Name, select R_Minimum Services.
4. In the Session Type definition, define the Priority for sessions of this type and the Minimum Services (minimum number of CPU slots required for sessions of this type).

   The minimum number of slots remains allocated to the session regardless of workload or priority of other sessions.

   The priority value is used to allocate service instances other than the minimum number of service instances.

For example, you have 66 service instances and three session types, and you defined the minimum number of instances to be two per session type.

Two instances are allocated to each session to meet the minimum instance requirement. Then, additional instances are allocated proportionally based on priority.

Session and Session Type	Minimum service instances configured	Priority	Allocated intances	
			Allocated instances (minimum)	Allocated instances (additional)
Session1, SessionA	2	10	2	10
Session2, SessionB	2	20	2	20
Session3, SessionC	2	30	2	30

The sessions receive two service instances each. The remaining 60 service instances are distributed to the sessions proportionally based on priority of the session type.

Session 1 gets 12 service instances in total, 2 gets 22 service instances, and session 3 gets 32 service instances.

5. Click Save to apply your changes.

Configuration to save historical data

Historical data comprises information related to completed tasks and sessions. Typically, historical data is stored in a cluster shared directory so that it is accessible to both Symphony Session Manager (SSM) and Session Director (SD). If this is the case, you can view historical data from the Console or from the command line without any further configuration.

If your cluster does not have a shared directory configured and SD and SSM are running on different hosts, you need to configure your application profile to enable access to the historical data. When you configure a location to store the historical data, you must ensure that it is accessible to both SD and SSM.

Configure the location for historical data files:

1. Open the application profile in an editor.
2. In the SOAM>DataHistory element, configure the path attribute. For example:

```
<SOAM version="5.1">
 ...
     <DataHistory fileSwitchSize="100" lastingPeriod="96"
     path="\\filerA\soam\work\history"/>
 ...
</SOAM>
```

   You can configure an absolute path or a relative path. If a relative path is specified, it is relative to `%EGO_CONFDIR%\..\..\soam\work` (Windows) or `$EGO_CONFDIR/../../soam/work (Linux)`.

Change the application profile to only log error historical data

By default, the system displays all data so that you can analyze which tasks have completed when developing client and services.

For performance reasons, you may want to configure your production environment to only enable historical data for tasks in the error state.

Note:

If you are manually editing the application profile outside the Platform Management Console, the parameters that you need to change are in the session type and are called `persistTaskHistory`, `persistSessionHistory`. Valid values are all, error, none.

1. In the Platform Management Console, click Symphony Workload > Configure Applications.

   The Applications page displays.
2. Select the application to modify.

   The Application Profile is displayed.
3. Under Session Type Definition, select a value for Logging History

Setting	Behavior
Log all sessions, error tasks only	Save all session history. Save task history only for those tasks that have completed in error.
Log all sessions, all tasks	Save all session history. Save task history for tasks in all states.
Log all sessions, no tasks	Save all session history. Do not save any task history.
Log no sessions, no tasks	Save no history at all.

4. Click Save to apply your changes.

CHAPTER

20

Application Error Handling

Feature: Host blocking

Host blocking—a feature of application error handling—prevents Symphony from repeatedly trying to run a service on a host that does not have adequate hardware or software resources. You can configure host blocking to take effect on timeout or exit for each of your services, or when a service throws an exception or sends a specific return code.

About host blocking

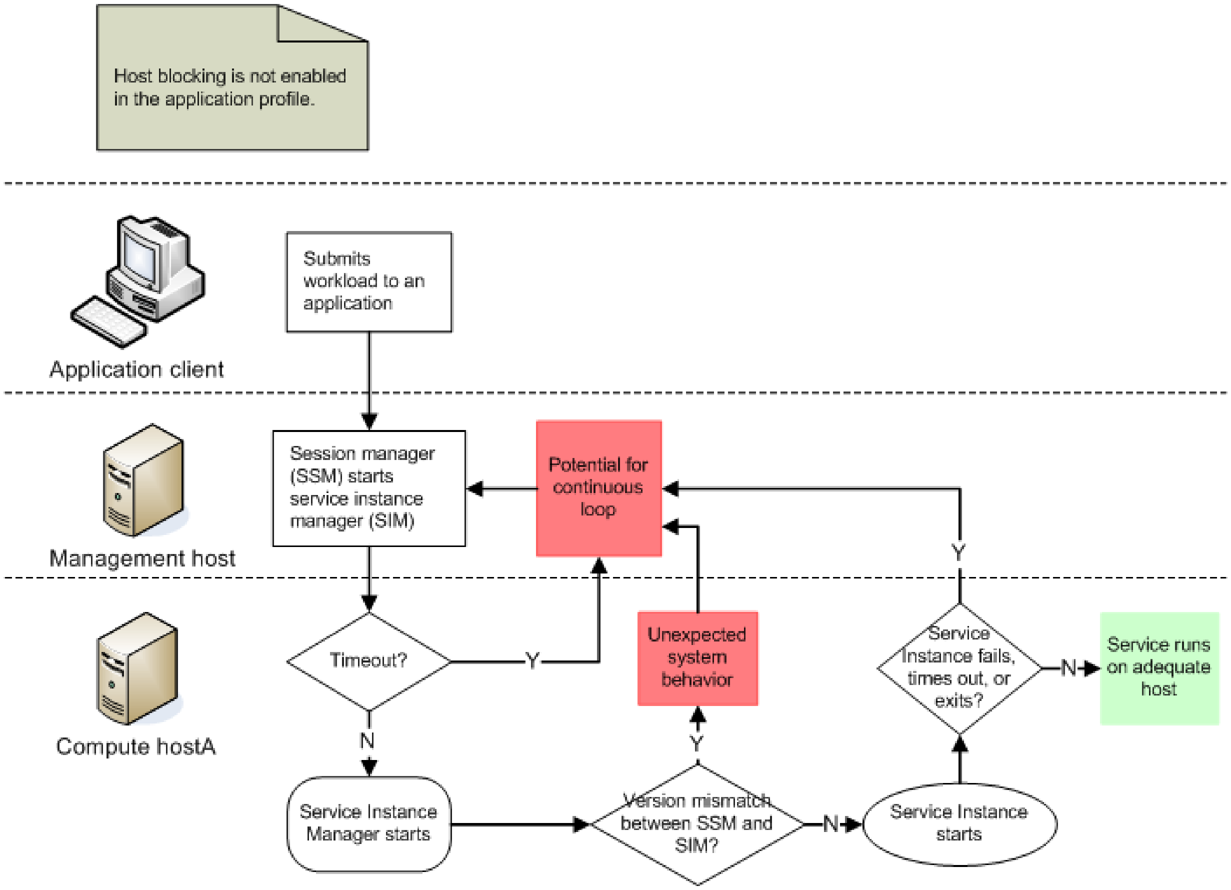
When host blocking takes effect, Symphony creates a blocked host list for the application with which the service is associated. A host that appears on the blocked host list can no longer be used by the application until you intentionally unblock the host, or the application is re-registered or disabled and enabled again.

By default, host blocking is enabled for a version mismatch or communication timeout between the session manager and the service instance manager. You can also configure host blocking for a service instance error, a service instance exit, or a service instance method timeout. By default, host blocking is enabled for the following service instance methods:

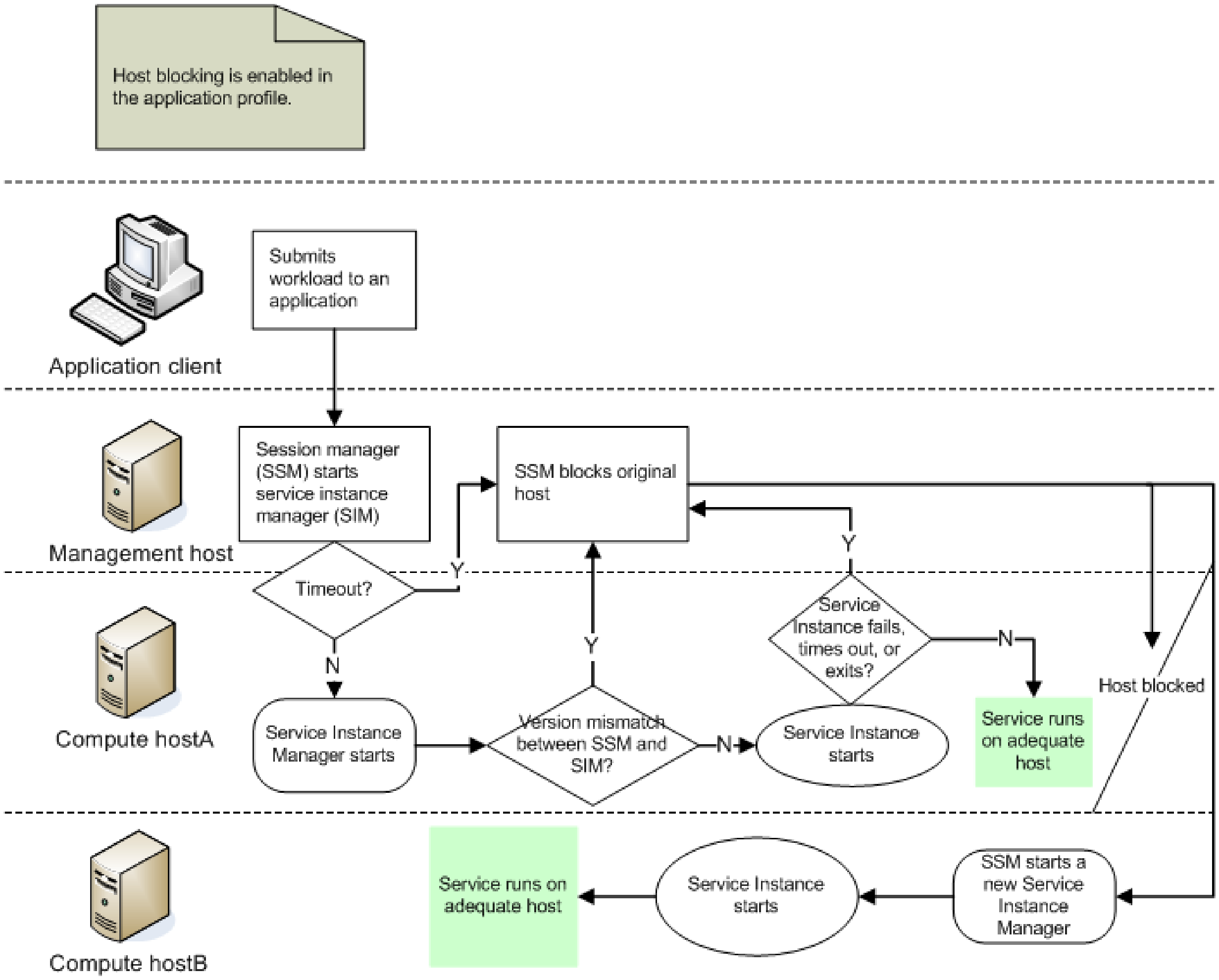
Method	**Event types**
Register	• Timeout • Exit
CreateService	• Timeout • Exit • Failure exception • Fatal exception
SessionEnter	• Timeout • Exit
SessionUpdate	• Timeout • Exit

The following illustrations show the benefits of using the host blocking feature.

Without host blocking (feature disabled)


Host blocking is not enabled in the application profile.
Application client
Submits workload to an application
Management host
Session manager (SSM) starts service instance manager (SIM)
Potential for continuous loop
Y
Compute hostA
Timeout?
Y
N
Unexpected system behavior
Y
Service Instance fails, times out, or exits?
N
Service runs on adequate host
Service Instance Manager starts
Version mismatch between SSM and SIM?
N
Service Instance starts

With host blocking enabled


Host blocking is enabled in the application profile.
Application client
Submits workload to an application
Management host
Session manager (SSM) starts service instance manager (SIM)
SSM blocks original host
Timeout?
Y
N
Compute hostA
Service Instance Manager starts
Version mismatch between SSM and SIM?
Y
N
Service Instance starts
Service Instance fails, times out, or exits?
Y
N
Service runs on adequate host
Host blocked
Compute hostB
Service runs on adequate host
Service Instance starts
SSM starts a new Service Instance Manager

Host blocking triggers

Host blocking triggers automatically when the session manager version on the management host does not match the service instance manager version on the comput host.

You can configure additional host blocking based on the requirements of your application so that Symphony triggers host blocking for any of the following reasons:

- A service method times out, exits or crashes, throws an exception, or returns certain control codes.
- The service instance manager does not communicate with the session manager before the configured timeout period expires (controlled by the startUpTimeout value).
- The service instance does not communicate with the service instance manager before the configured timeout period expires (controlled by the setting for the `Register` method actionOnSI attribute).

Slot blocking for Symphony DE

Symphony DE blocks slots—not hosts—under the same conditions that trigger host blocking for a production grid. Symptoms of blocked slots include fewer resources than expected or no resources serving your application, more tasks in the `PENDING` state, a slower rate of workload completion, and clients that hang. You can check for blocked slots by looking in the `ssm.`*`hostname.app_name`*`.log` file and searching for `WARN` or `ERROR` messages about blocked hosts. If you see a blocked host message, one or more slots might be blocked. You can unblock slots by disabling and then enabling the application or by restarting the DE cluster.

Scope

Applicability	Details
Operating system	• All host types that are supported by the Symphony system.
Limitations	• For Symphony DE, only slots are blocked.

Configuration to enable host blocking

Host blocking is enabled in the application profile for each application. You can configure host blocking at the service instance manager level, the service instance level, or both.

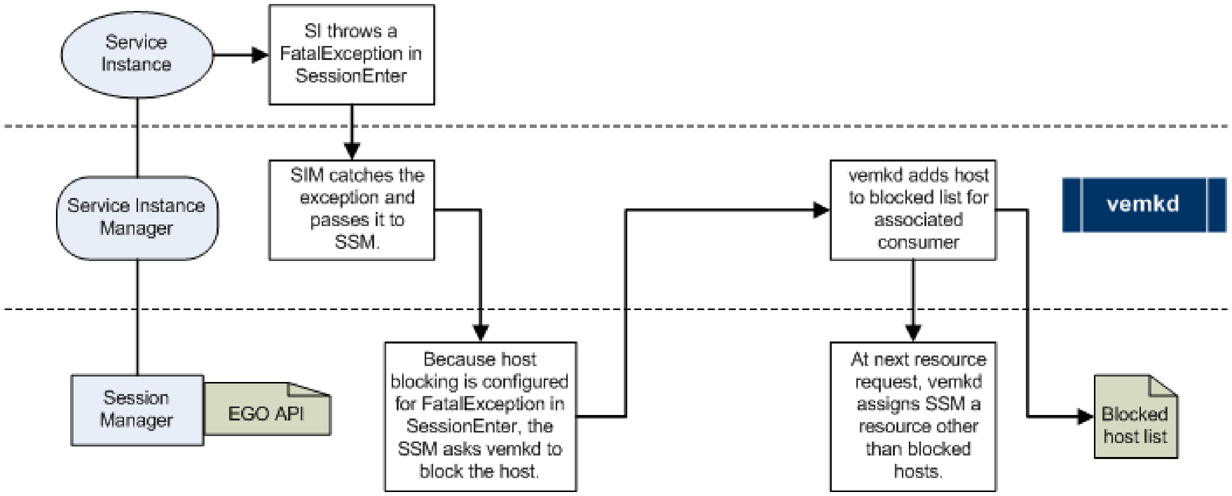
Section	Attribute name and syntax	Behavior
SOAM > SIM	blockHostOnTimeout="true"	• Enables host blocking for the application when the service instance manager times out while trying to communicate with the session manager. • Used with the startUpTimeout attribute.
	startUpTimeout="*seconds*"	• Number of seconds to wait for the service instance manager to communicate with the session manager. This attribute works in conjunction with blockHostOnTimeout. • When the process times out, the session manager requests a new host from EGO and tries to start a new service instance manager on the new host.
Service > Control > Method > Timeout	actionOnSI=blockHost	• When a timeout is reached on the method, terminates the running service instance on this host and does not use this host to start any other service instance for the application. • Used with the duration attribute. • You can specify the blockHost option for the following methods: • Register • CreateService • SessionEnter • SessionUpdate • Invoke • SessionLeave

Section	Attribute name and syntax	Behavior
Service > Control > Method > Exit	actionOnSI=blockHost	- When the service instance exits or crashes during execution of the method, the system does not use this host to start any other service instance for the application - You can specify the blockHost option for the following methods: - Register - CreateService - SessionEnter - SessionUpdate - Invoke - SessionLeave
Service > Control > Method > Return	actionOnSI=blockHost	- When the method returns normally or with a specified control code, terminates the running service instance on this host and does not use this host to start any other service instance for the application. - You can specify the blockHost option for the following methods: - CreateService - SessionEnter - SessionUpdate - Invoke - SessionLeave
Service > Control > Method > Exception	actionOnSI=blockHost	- When the specified exception (failure or fatal exception) occurs, terminates the running service instance on this host and does not use this host to start any other service instance for the application. - You can specify the blockHost option for the following methods: - CreateService - SessionEnter - SessionUpdate - Invoke - SessionLeave

Host blocking behavior

When host blocking is triggered, the system creates a blocked host list for the application. The following example illustrates the host blocking process triggered at the service instance level.

Example of the host blocking process


Service Instance
SI throws a FatalException in SessionEnter
Service Instance Manager
SIM catches the exception and passes it to SSM.
vemkd adds host to blocked list for associated consumer
vemkd
Session Manager
EGO API
Because host blocking is configured for FatalException in SessionEnter, the SSM asks vemkd to block the host.
At next resource request, vemkd assigns SSM a resource other than blocked hosts.
Blocked host list

Configuration to modify host blocking behavior

Not applicable. There are no attributes that change the way that host blocking works other than those attributes configured in the application profile.

Host blocking actions

Actions to monitor

You can monitor host blocking through the Platform Management Console (PMC), the command line, and through the Symphony log files located in the `logs` directory of `SOAM_HOME`. You can also trap SNMP events to receive notifications when a service triggers the system to block a host.

User	Action	Description
- Cluster administrator	From the Platform Management Console: **Symphony Workload > Monitor Workload > *application_name* > Blocked Hosts**	- Displays a list of blocked hosts for the selected application.
- Cluster administrator - Consumer administrator	From the command line: `egosh alloc view`	- 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.

You can find information about host blocking in the following log file:

Log file	Location	Event	Description

Session manager log file: `ssm.`*`host_name`*`.`*`app_name`*`.log`	Linux/UNIX:
	`$SOAM_HOME/logs`
	Windows:
	`%SOAM_HOME%\logs`
		`SOA_SERVICE_BLOCKED`	Error level message that indicates that host blocking has occurred.

Actions to control

Typically, a cluster administrator removes a blocked host when the host has been modified—by means of a software or hardware upgrade, for example—to meet the requirements of the service. A host can be removed from the blocked host lists in one of two ways:

- Directly from the Platform Management Console (PMC)
- Indirectly, by disabling and re-enabling the application associated with the blocked host

User	Action	Behavior
• Cluster administrator	From the Platform Management Console: **Symphony Workload > Monitor Workload > *application_name* > Blocked Hosts > *host_name* > Actions > Remove from Blocked Hosts**	• The system removes the host from the blocked host list • The application can start a service on the previously blocked host
• Cluster administrator	From the Platform Management Console: **Symphony Workload > Configure Applications > *consumer_name* > *application_name* > Actions > Disable**	• Disables the application, which clears the blocked host list for the disabled application • No clients can be served by the disabled application
• Cluster administrator • Consumer administrator • Consumer user	From the command line: `soamcontrol app disable` *application_name*	• Disables the application, which clears the blocked host list for the disabled application • No clients can be served by the disabled application • For information about how to use the `soamcontrol` command to disable and enable applications, see the *Reference*
• Cluster administrator • Consumer administrator	From the Platform Management Console: • **Symphony Workload > Configure Applications > *application_name* > Basic Configuration > Save** • **Symphony Workload > Configure Applications > *application_name* > Advanced Configuration > Save**	• The system first disables and then re-registers the application, which clears the blocked host list for the modified application

For Symphony DE, you can unblock slots by disabling and then enabling the application, or by restarting the DE cluster.

User	Action	Behavior
• Developer	From the command line: `soamcontrol app disable` *application_name*	• Disables the application, which unblocks slots for the disabled application • No clients can be served by the disabled application
	`soamcontrol app enable` *application_name*	• Enables the application, which can start services on any previously blocked slot
• Developer	Windows: • Right-click on the Symphony DE tray icon and choose **Stop Symphony DE on all hosts**. Once the DE cluster shuts down, right-click on the Symphony DE tray icon and choose **Start Symphony DE on all hosts**. Linux/UNIX: • `soamshutdown` • `soamstartup`	• Shuts down and then restarts Symphony DE • Unblocks slots for all applications running on the DE cluster

Actions to display configuration

User	Command	Behavior
• Cluster administrator • Consumer administrator	From the Platform Management Console: • **Symphony Workload > Configure Applications >** ***application_name*** **> Basic Configuration** • **Symphony Workload > Configure Applications >** ***application_name*** **> Advanced Configuration**	• Displays application profile settings for the selected application
• Cluster administrator • Consumer administrator • Consumer user	From the command line: • `soamview app` *app_name* `-p`	• Displays application profile settings for the selected application

You can also view an application profile using an XML editor.

CHAPTER

21

Application Tuning

Using standby services to reduce service startup times

Standby services minimize the need to restart services at the time resources are allocated to an application by allowing these services to run idle when there is no workload.

Scope

Applicability	Details
Operating system	• Windows • Linux • Solaris
Limitations	The standby service feature is not supported by Symphony DE. In this case, standby service configuration is ignored by Symphony.

About standby services

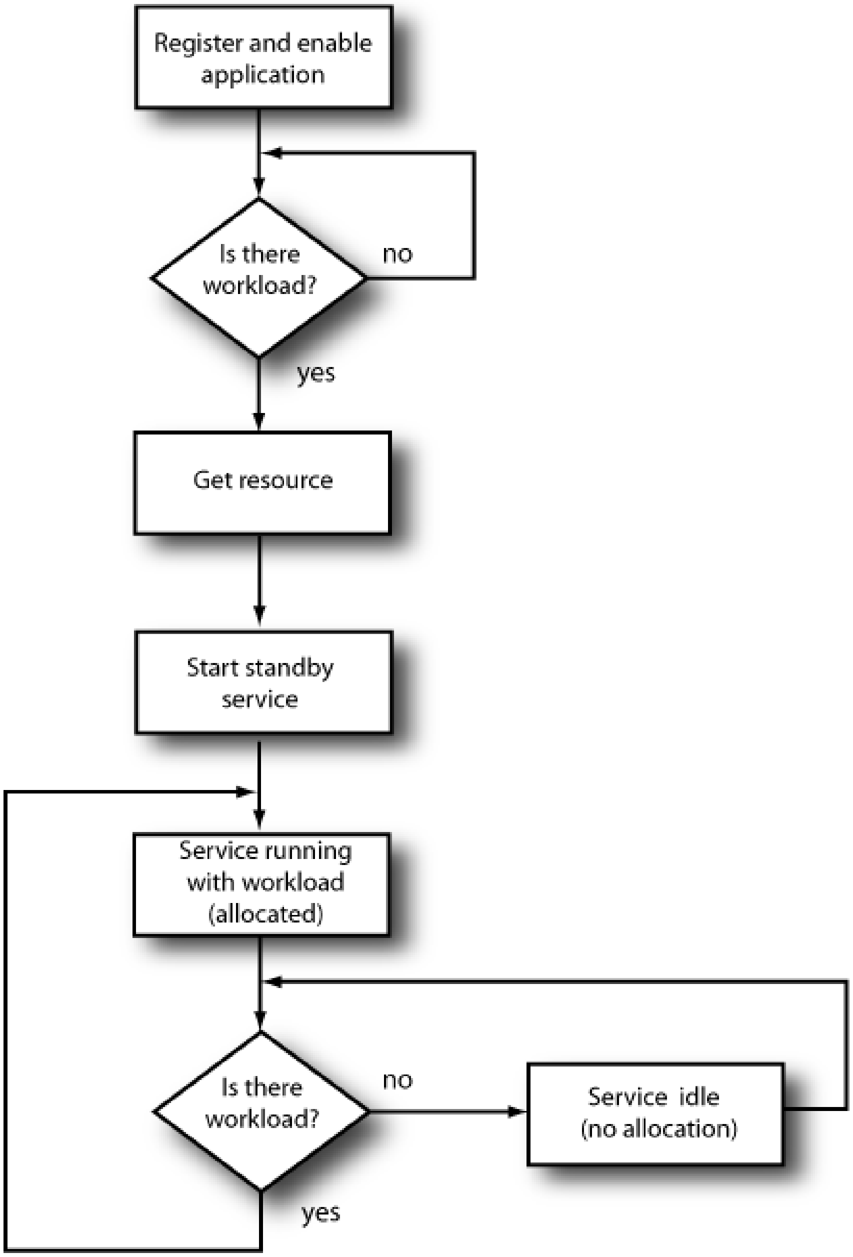
To maximize the utilization of resources, Symphony, by default, releases resources as soon as there are no tasks pending. Each time the resources are released, service instances on these resources are terminated. When more tasks are received, new resources are allocated and the services start again. Sometimes starting up a service takes much longer than the actual run time of the workload; for time-critical workload, this may not be acceptable.

Standby services minimize the need to restart services when resources are allocated to an application by allowing these services to keep running. Standby services also allow other consumers to use these resources when there is no workload for the running service. This is due to the fact that standby services do not occupy slots, thereby allowing EGO to allocate these resources to other applications. Once the service instance is associated with a slot and is used to run tasks, it is no longer considered a standby service.

Standby services are only started on a resource when the application has workload to process. So it is possible that even though all the resources in a resource group are configured to run standby services, not all resources may have them running. Once the standby service is started, it remains running until the application is unregistered or disabled.

System behavior when applications are configured with standby services

This section describes Symphony behavior during the lifecycle of a standby service.


Register and enable application
Is there workload?
no
yes
Get resource
Start standby service
Service running with workload (allocated)
Is there workload?
no
Service idle (no allocation)
yes

1. The lifecycle begins when an application is registered and enabled.
2. The Session Director reads the application profile and starts the Session Manager (SSM) for the application.
3. When the SSM receives workload for the application, the SSM requests resources from EGO.
4. For each resource received, the SSM sends the service information to the SIM and the SIM starts the standby service instance.
5. When the workload is finished, the SSM releases the slot to EGO but keeps the standby activity alive for the service. For applications with multiple services, the standby activity for the default service is kept alive.
6. EGO unallocates the slot but keeps the standby service running.

7. If the SSM receives new workload, EGO allocates the resources that have the standby services running.
8. When the SSM receives the resource allocations from EGO, it associates the resource with the SIM already running on the resource, rather than start a new SIM. The activities are reassociated with the allocations.
9. The standby services are shut down when the application is disabled or unregistered.

System behavior when applications with standby services share resources

When Symphony is optimized for standby services, EGO allocates resources to applications by first searching through the resource groups that are dedicated to standby services. Therefore, it is recommended that the consumer own all the resources in a resource group dedicated to standby services, as it guarantees that the resources with standby services running will be available to the consumer when they are needed. If the consumer has unsatisfied demand and it previously lent out resources with standby services to another consumer, EGO will reclaim them before allocating an idle resource. In this case, workload will be pending until the resource is reclaimed. However, if no standby service is running on the resources lent out, EGO will allocate an idle resource before reclaiming the resources that were lent out.

Note that if the reclamation period is longer than the service start-up time, the optimization for standby services will not be beneficial. In this case, it would be better to disable the standby service optimization so that EGO allocates any idle resource to the application.

To enable reclaim optimization for standby services, select the Optimized for standby service setting using the PMC.

Failure recovery

This section describes system behavior pertaining to the recovery of standby services in the event of a Symphony component failure.

EGO failure

After recovery, all the information related to standby services is restored by EGO. All the allocations and activities will be recovered including the activities without slots allocated

SIM failure

If the SSM detects a standby SIM failure, the number of slots with standby services in the system decreases by the number of slots affected by the failure. There is no request to restart the standby SIM immediately. When workload is submitted, the SSM requests the necessary resources and then consumes the ones with standby services first. If the resources demanded by the workload consume all the existing standby services within the system, the SSM requests EGO to start new SIMs, which start new serv ices. After the workload completes, the SSM returns the resources and keeps the SIMs and their service instances in standby mode.

Standby service failure

Since the SIM does not monitor the Service Instance while it is idle, if the standby service goes out of service, the SIM will not know it. When the SIM is assigned to a session that wants to use the standby service, the SIM must start a new service before it can submit tasks to it.

Standby services and preloaded services - what’s the difference?

Symphony offers two ways to handle services with long start-up times: standby services and preloaded services. Preloaded services are started before workload is submitted by the client and require the resource to remain allocated to the application. Consequently, the resource cannot be shared with other consumers. A resource with standby services running, on the other hand, is only allocated to an application when there is workload to be processed. Once workload is finished, the resource is released (with the service running in standby mode), and made available to other consumers.

Standby services can be combined with preloaded services. In this case, an application can be configured to have a number of slots for preloaded services in addition to standby services. When the application receives workload, if the number of pre-loaded services cannot satisfy the demand, the SSM requests new resources from EGO and uses the standby services to supplement the requested resources.

When to use standby services

Standby services are recommended for environments where the tasks are sent intermittently and the service startup time is relatively long in comparison to the running time of the tasks. By using this feature, users can reduce the impact of starting services on the overall task turnaround time.

Here are additional considerations when deciding if standby services are the best choice:

- Since standby services are kept running and they occupy host resources such as memory, they are not recommended for services with nominal startup times or services that are memory-intensive, which makes the host unusable by other applications. In this case, it is recommended to use preloaded services to retain the resources and keep the services running.
- Does this application's resource plan entitle it to own (ownership) or deserve (share ratio) a number of slots? If the answer is no, then it is not suitable for standby services.
- Do other applications need the selective reclaim feature? If the answer is yes, do not use standby services as standby service configuration will not allow selective reclaim to take effect.

Configuring standby services: best practices

You should follow the best practices outlined here to ensure that resources with standby services running are available when required by the application.

1. Create a resource group exclusively for the application’s standby services.
2. The application’s consumer should own all the resources in the resource group.
3. Enable lending and disable borrowing in the consumer’s resource plan. Borrowing would not guarantee the availability of a resource with the standby service running when EGO allocates the resource to the application.

Configure standby services

Standby services are configured in the application profile. The following steps describe configuration using the Platform Management Console (PMC). To configure standby services manually, set the enableStandbyServices attribute to true in the Consumer section of the application profile.

1. In the PMC, click Cluster > Summary > Cluster Properties.
2. Under the Reclaim optimization heading, select Optimized for standby service. Click Apply.

Important:

In some cases, optimizing for standby services may not produce the expected results; refer to the Cluster and Application Management guide for more information about sharing resources with standby services.

3. Click Symphony Workload > Configure Applications.

   The Applications page displays.

4. Select the application you want to modify.

   The Application Profile page displays.
5. In the General Settings section, if you want to include preloaded service instances in your application, enter the number of slots in the Number of slots for preloaded services textbox.
6. In the Enable standby service dropdown list, select true. This ensures that resources allocated to the application will have standby services running when slots are released by the SSM.
7. In the Slots threshold for standby services textbox, enter the maximum number of slots that standby services can run on for this application.
8. Click Save to apply your changes.

   Symphony reregisters the application for the changes to take effect. The standby services will be started and remain running once workload is submitted to the application.

Limit the number of service instances that can run on a host

Use this feature to limit the number of service instances that an application can run on a host; for example, you have an application that consumes a lot of memory and you want to limit the amount of memory that the application consumes on a host.

This feature uses the configured number of resources on a host to limit the number of service instances that can run on the host. A host resource can represent almost anything installed on a host such as a software license, a graphics accelerator, etc., or it can represent a virtual resource. By specifying the number of resources on a host, you can effectively control the number of service instances that an application can run on that host. When an application needs the resources, EGO does not distribute slots to it from a host unless there are sufficient available resources on that host.

In essence, this feature allows a particluar type of application to run only N service instances on a host with M slots where M >=N. The remaining slots can be used by other applications.

Here is an example;

- A cluster is made up of 25 compute hosts. Each compute host is configured to have 4 resources.
- Application1 is configured to need 2 resources to run each service instance.
- Application2 is configured to need 1 resource to run each service instance.

100 long-running tasks are submitted for Application1 and Application2.

- On the hosts that only run tasks for Application1, there will be at most 2 tasks.
- On the hosts that only run tasks for Application2, there will be at most 4 tasks.
- On the hosts that run tasks for both Application1 and Application2, there will be 1 task for Application1 and 2 tasks for Application2.

Feature interactions

This feature is not compatible with the exclusive slot allocation policy; refer to *Exclusive slot allocation policy* on page 187.

Configuration

Configure the resource name

Open the `ego.shared` file. In the Resource section, specify a name for each type of resource you want to configure.

For example:

```
Begin Resource
RESOURCENAME   TYPE    INTERVAL INCREASING  DESCRIPTION  # Keywords
   ...
limited_res  Numeric ()     N       (the number of resources on a host )    ...
End Resource
```

The first line consists of the keywords. `RESOURCENAME` and `DESCRIPTION` are mandatory. Keyword `TYPE` must be set to **Numeric** for resources used by this feature. The other keywords are optional. Subsequent lines define resources.

Configure the number of host resources

You can either specify the number of available resources on specific hosts or you can set the number of resources as a global default for all hosts in the cluster. The global default also applies to any dynamically added hosts whereas the host-level configuration only applies to static hosts. Note that these two methods of configuration are mutually exclusive and only one method can be used at one time for the same resource.

1. Open the `ego.cluster` file. Do one of the following:
   - To configure the number of resources on specific hosts, follow this step:

     In the Host section, specify the host name, the resource name, and the number of resources available on the host.

     For example:

     ```
     Begin Host
     HOSTNAME  model    type        r1m  mem  swp  RESOURCES    #Keywords
     HostA !     !      -   -   -  (limited_res=5)
     HostB !     !      -   -   -  (limited_res=3)
     End     Host
     ```
   - To configure the number of resources as a global default for all hosts in the cluster, follow this step:

     In the ResourceMap section, specify the resource name, and the default number of resources available on every host.

     For example:

     ```
     Begin ResourceMap
      RESOURCENAME  LOCATION
      ...
      limited_res          1@[default]
      ...
      End ResourceMap
     ```
2. Open the application profile. In the consumer section, configure the number of resources each service will consume on each host you specified in the `ego.cluster` file.

   For example:

   ```
   ...
   <Consumer applicationName="newApp" resReq="rusage(limited_res=2)" ...>
   ...
   ```

   In this example, the application needs 2 resources to run each service instance.

View the host resources

You can view a list of resources configured for each host using the CLI. For example, to view the "limited_res" resources, run the following command:

egosh resource list -o limited_res

Resource-aware scheduling for sessions

This feature provides a means for the application developer to state a resource preference for host attributes that provide the best match for the workload in a session.

Scope

Applicability	**Details**
Operating system	• Windows • Linux • Solaris
Limitations	Not applicable to Symphony DE

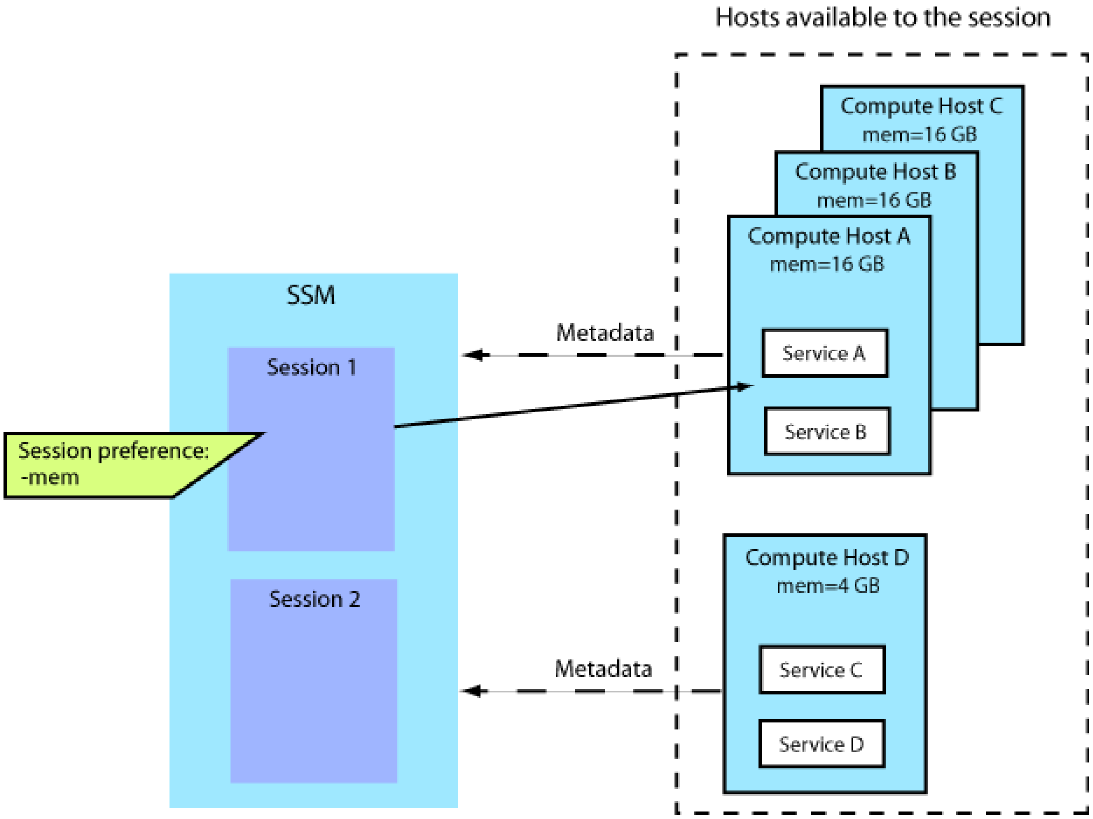
About resource preference

In a typical cluster environment, all machines are not identical. Some machines may be slower and some machines may have less memory than others. If you configure a resource requirement at the application level, all sessions for that application have to use the same set of resources. However, it may not be the case that all sessions are the best match for the resources. Some sessions may benefit from special resources so that results can be computed in the most optimal way.

With the resource preference feature you can specify a preferential association between a session and a host that possesses the attributes best suited to process the workload. The association is based on the evaluation of a user-defined expression containing resource attributes capable of being monitored. The evaluation of this expression is carried out against the attributes of each host available to the session. Typically, a resource attribute may be a preferred attribute of host machines that you want the service to be running on when it processes the workload. These resource attributes are made available to the middleware on a continuous basis whether or not there are sessions that make use of this information.

When you state a resource preference, there is no guarantee that you will receive the hosts with the preferred attributes; this depends on the resources available at the time the workload is being scheduled.

The following example illustrates the concept of resource preference for sessions. Resources are assigned to the session based on the resource preference at the session level. In this case, it is determined that session 1 prefers to run on hosts with the most memory (-mem). Host A, B, and C have 16 GB memory while host D has 4 GB. The SSM collects metadata (host attributes) from all the resources available to session1 at that moment and selects Host A, B and C since they represent the best match for the session according to the specified preference. Note that if session 2 had been created before session 1 and it had no resource preference, any of the four hosts could have been allocated to session 2. In that case, session 1 would only have the three remaining hosts available.



How preference is considered

When no preference is specified, there will not be any preference evaluation when assigning a service to a session. Once a service is assigned, tasks will be dispatched in the usual manner.

When a preference is specified, there are two opportunities for the preference evaluation to be done:

1. Upon initial assignment of an idle or new host. In this case, preferences are evaluated before ordering the available hosts for assignment. Services that have been pre-started, or in standby and idle will most likely have enough information collected to make the best assignment decision.
2. Reassignment of services due to rebalancing of resources. In this case, if a service can be reassigned to multiple sessions, an evaluation can be done to determine which session the service has the best affinity with.

Expressing resource preference

Specifying attributes in a preference expression

Attributes take the form of identifiers and can be applied with the +, -, *, and / operators within an expression. An example of an expression would be "-mem", meaning that it is preferred to send workload to a host that possesses the most memory. The “/” and “*” operators can be used to normalize the preference string if you are setting preference for disparate types of attributes such as mem and cpu factor in the same expression. The “- “operator can be used to reverse the direction of the attribute. For example, since the best preference is the expression that resolves to the lowest value then if you want the most memory you can use the “-“ to reverse the direction so that the hosts with the most memory will have the most negative value and hence will be more favorable.

Attribute names can only contain alphanumeric and underscore characters; if you want to use resource attributes with names that do not comply with these rules, aliases must be defined. A resource attribute definition can be used to define an alias and to override the default value for an attribute should the session level default or system default be inappropriate.

The following out-of-the-box attributes are related to the host the service is running on and are available for use in a resource preference expression. See the Load Indices topic in the Symphony Reference guide for details.

Attribute	Description	Range	Preference Representation
Run queue length	The number of processes running on the host over the last 15 seconds	0.0 - MAX Float	r15s
Run queue length	The number of processes running on the host over the last minute.	0.0 - MAX Float	r1m
Run queue length	The number of processes running on the host over the last 15 minutes.	0.0 - MAX Float	r15m
CPU utilization	The percent of the overall processing power that is currently being used on the host averaged over the last minute.	0.0 - 1.0	ut
Paging activity	The number of pages being swapped in and out of memory per second averaged over the last minute.	0.0 - MAX Float	pg
Logins	The number of users logged into the host.	0.0 - MAX Float	ls
Idle time	The amount of time the host been idle for.	0.0 - MAX Float	it
Available swap space	The number of megabytes of swap space available on the host.	0.0 - MAX Float	swp
Available memory	The number of megabytes of physical memory available on the host.	0.0 - MAX Float	mem
Available temporary file space	The number of megabytes of disk space available in the temporary file directory.	0.0 - MAX Float	tmp
I/O activity	The number of kilobytes being written to disk per second averaged over the last minute.	0.0 - MAX Float	io

How Symphony handles the result of the expression

The result of each expression is a numeric value that is obtained by applying the operators to the attributes in the expression. Once the result is obtained for each resource being evaluated, it is be used to sort the resources in ascending order. This means the resource that evaluates to the lowest value is the most preferred.

When no information is available for a resource attribute that is involved in a resource evaluation, the resolution of the expression still proceeds. In such cases, Symphony attempts to substitute a default value for each attribute that it cannot resolve. The value of the attribute is resolved by the system in the following manner, in the given order:

1. attempt to find any collected value

2. retrieve the configured default for the session (if defined)
3. retrieve the system default (1.0E+300)

Configuring resource preference

Enabling resource preference for sessions

Resource preference is enabled at the application level with the schedulingAffinity attribute in the Consumer element of the application profile. When the attribute is set to ResourceAware, metadata is collected by the SSM and the resource preference is applied.

Example:

```
<Consumer applicationName="SharingDataCPP" ...schedulingAffinity="ResourceAware" />
```

The schedulingAffinity attribute can be configured through the PMC or by manually editing the application profile.

Configuring resource preference for sessions

Resource preference is configured in the SessionTypes element of the application profile. The preference expression is specified with the preference attribute. The default value for the preference expression, in the event that the value of the attribute in the expression is not known, is configured with the defaultResourceAttributeValue attribute.

Example:

```
<SessionTypes>
<Type name="typeA" ... preference="-mem"
defaultResourceAttributeValue="1.0" />
</SessionTypes>
```

The preference and defaultResourceAttributeValue attributes can be configured through the PMC or by manually editing the application profile.

Client API

Resource preference for sessions can be configured using the client-side API. This configuration overrides the setting in the application profile. For more information about the API, refer to the API reference documentation.

Defining default attribute values for the session

You can define a substitute value for attributes in a preference expression in the event that those attributes are unknown at the time of evaluation. If this default value is not specified, the system default, i.e., the 1.0E+300 value, is substituted.

Example:

```
SessionCreationAttributes.setDefaultResourceAttributeValue (float)
```

Specifying resource preference

You can programmatically describe any preference for resources.

Example:

```
ResourcePreference rp ("-mem")
```

This expression means that it is preferred to run this workload on a host that has the most memory.

Associating a resource preference with a session

A resource preference is associated with a session by binding it to a SessionCreationAttributes object.

Example:

```
ResourcePreference rp (“-mem”);
SessionCreationAttributes sessionAttributes();
sessionAttributes.setResourcePreference(rp)
```

Configure resource preference

Resource preference is configured in the application profile. The following steps describe configuration using the Platform Management Console (PMC). To configure a resource preference manually, set the appropriate attributes in the application profile as described in the resource preference feature reference.

1. In the PMC, click Symphony Workload > Configure Applications.

   The Applications page displays.
2. Select the application you want to modify.

   The Application Profile page displays.
3. Expand the Resource and Data Aware Scheduling section. In the Scheduling Affinity dropdown list, select Resource Aware.
4. Expand the Session Type Definition section. In the Resource Preference textbox, enter the host attribute(s) that you want to include in the preference expression.
5. In the Default Resource Attribute Value textbox, enter a value that will be substituted by the middleware if the preferred host attribute value is not available.
6. Click Save to apply your changes.

   Symphony reregisters the application for the changes to take effect.

Optimizing common data for multi-slot hosts

The transfer of common data can be optimized for environments that feature sessions running multiple SIMs on the same host.

Scope

Applicability	Details
Operating system	• Windows • Linux • Solaris
Limitations	N/A

About common data and multi-slot hosts

When common data for a session is sent to compute hosts, Symphony, by default, sends one copy of the data to each SIM. In the case of one session running multiple SIMs on the same host, the same data will be sent multiple times. The potential drawback of having to send and maintain multiple copies of common data and common data updates is the burden it puts on host memory and network bandwidth.

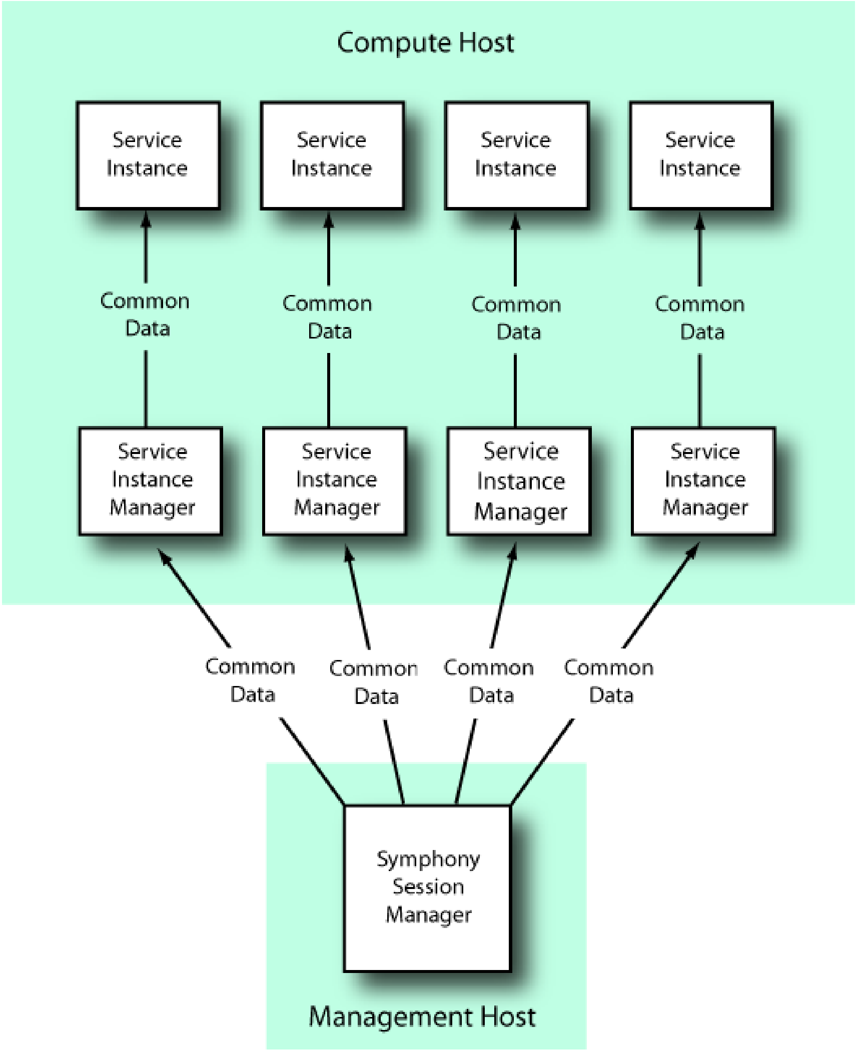


Figure 3: Common data distribution, Symphony default model

Common data optimization

Symphony allows you to optimize the distribution of common data and common data updates so that only one copy of the data is sent to each host serving the same session. The single copy of common data is shared among all SIMs and SIs associated with the same session on the same host. Note that if all the SIMs on the host get unbound from the session, the common data is lost and must be sent to the host again if the SIMs get bound to that same session.

Common data is shared among the SIMs using a shared file system. When the first SIM for the session is started, it receives the common data and stores it in a shared file. When additional SIMs are assigned to the same session, they use the common data stored in the shared file. The same principle applies to common data updates, which are stored in a shared file at the same location as the common data.

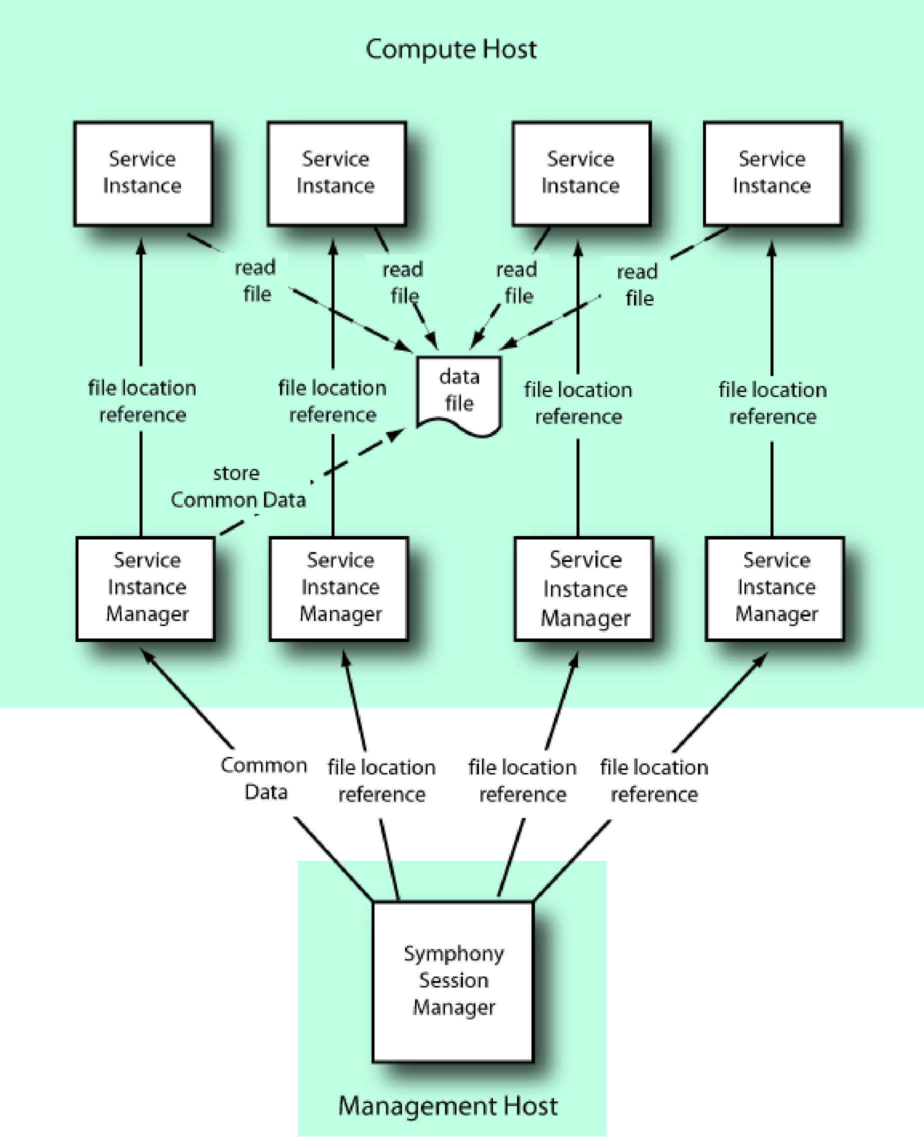


Figure 4: Common data distribution, optimized

When to use common data optimization

Common data optimization is recommended for sessions that are likely to use multiple SIMs on the same host. Also, the impact of common data optimization is proportional to the size of the common data. The greater the common data size, the greater the benefit of optimization. For example, if the common data size is 100 MB and the compute host has four SIMs assigned to the session, the session round trip can be completed with an 80% savings in time when compared to optimization being disabled. On the other hand, if the common data size is very small or the compute host has enough memory to store the multiple copies of common data, optimization may not offer any benefit.

Configuring common data optimization

Common data optimization is configured by setting the enableCommonDataOptimization attribute at the session level in the application profile. For example:

```
<SessionTypes>
```

```
        <Type name="ShortRunningTasks" priority="1" recoverable="false"
              sessionRetryLimit="3" taskRetryLimit="3"
              abortSessionIfTaskFail="false" abortSessionIfClientDisconnect="true"
              suspendGracePeriod="100"  taskCleanupPeriod="100"
enableCommonDataOptimization="true" />
    </SessionTypes>
```

Delaying the release of slots after running workload

You can configure the amount of time that an unassigned service will remain idle with the application before releasing its corresponding slot to EGO (and terminating the service). An unassigned service is defined as a service not assigned to any session. If new workload arrives before the configured time elapses, these unassigned services can be used to handle the new workload. Once an unassigned service becomes assigned to a session again, its idle timer is reset.

After a session is suspended or killed, if other outstanding workload does not use the service, the service will wait for the slot release delay time to expire before the resources are released back to EGO.

The slot release delay time can be configured for the whole application or for an individual service. Configuration for a service overrides the application-level configuration.

Feature interactions

This section summarizes the behavior of the slot release delay feature when interacting with other Symphony features.

taskLowWaterMark

This feature only applies when taskLowWaterMark = 1.0 since a taskLowWaterMark value of 0 means that the SSM never releases resources voluntarily.

Exclusive host

When exclusive host is configured, an SSM must release all of the slots on the host back to EGO at once. All services on the host will be terminated (and the slots released) when all services are idle and the slot release delay time has expired for all of the services on that host.

Preloaded services

When an application is configured with preloaded services, the SSM tries to retain the configured number of slots (numOfSlotsForPreloadedServices). The slot release delay feature only applies to slots in excess of the number of preloaded slots.

Standby services

The slot release delay feature can be used in conjunction with standby services. The unassigned service will remain idle with the application for the configured duration before becoming a standby service. An unassigned service in this idle state is in a better state of readiness than a standby service since it is not necessary for the SSM to make a resource request and wait for EGO to allocate the resource back to the SSM.

Dynamic services

An SSM cannot release a partial slot back to EGO; it must release the whole slot. If partial slots are configured for a service, the SSM must wait to find at least one slot worth of services whose slot release delay time has expired before it can release the slot.

Configuring slot release delay

Slot release delay is configured by setting the delaySlotRelease attribute at the application or service level in the application profile. For example:

```
<Consumer applicationName="symping" consumerId="/SymTesting/Symping"
delaySlotRelease="8" .../>
```

```
...
<Service default="true" delaySlotRelease="4" description=""... >
```

Optimizing session manager performance

Flow control

Flow control prevents session manager from exhausting critical system resources, which may occur under extreme workload.

Flow control does the following:

- Monitors the status of system resources for session manager:
  - Available virtual memory (physical plus swap memory available on the host)
  - Available virtual address space
  - Process memory (physical memory available for each process)
- Raises events when a certain threshold has been reached:
  - NORMAL - Operate in default mode for any new input
  - PROACTIVE - Gives early warning to system components that can make memory available when required
  - SEVERE - Starts to scavenge as much memory as possible, current clients work fine
  - CRITICAL -Starts to slow data inflow to the session manager and raise the priority of getting data out of the session manager. Rejects new connections, suspends new sessions from currently attached clients, and pends new tasks in those suspended sessions—current sessions and tasks work fine.
  - HALT - Session manager enters into lockdown mode, no further processing is allowed until an administrative action is performed, or the system enters a safer state.

1. Edit your application profile. In the SOAM SSM section, configure values for memory and virtual address space for each threshold.

   In the example below, when available memory on the session manager host is down to 50% of total memory, the event BEV_PROACTIVE is triggered.

   When available memory is down to 40%, the event BEV_SEVERE is triggered.

   For available virtual address space, when there is only 50% available virtual address space for the session manager process, BEV_PROACTIVE event is triggered.

   Similarly, for process memory, when there is only 40% available memory for the session manager process, BEV_PROACTIVE event is triggered. When available process memory is down to 30%, the event BEV_SEVERE is triggered, and so on.

```
<boundaryManagerConfig>
               <boundaries>
                   <boundary elementName="AvailableMemory">
                       <event name="BEV_PROACTIVE" value="50"/>
                       <event name="BEV_SEVERE" value="40"/>
                       <event name="BEV_CRITICAL" value="25"/>
                       <event name="BEV_HALT" value="15"/>
                   </boundary>
                   <boundary elementName="ProcessMemory">
                       <!-- MaxSizeReference will be capped at
                       "2048" by ssm when it runs on 32 bit OS
                       -->
                       <param name="MaxSizeReference"
                       value="8388608"/>
                       <event name="BEV_PROACTIVE" value="40"/>
                       <event name="BEV_SEVERE" value="30"/>
                       <event name="BEV_CRITICAL" value="20"/>
                       <event name="BEV_HALT" value="10"/>
                   </boundary>
```

```
                    <boundary elementName=
                        "AvailableVirtualAddressSpace">
                        <!-- MaxSizeReference will be capped at
                        "2048" by ssm when it runs on 32 bit OS
                        -->
                        <param name="MaxSizeReference"
                        value="8388608"/>
                        <event name="BEV_PROACTIVE" value="50"/>
                        <event name="BEV_SEVERE" value="40"/>
                        <event name="BEV_CRITICAL" value="25"/>
                        <event name="BEV_HALT" value="15"/>
                    </boundary>
                </boundaries>
            </boundaryManagerConfig>
```

2. Save your application profile.
3. Update your application with the new profile with the `soamreg` command. (If you prefer, you may do these steps using the Platform Management Console to export and import the application profile.)

Memory allocation parameters

If the SSM on Linux remains at a critical memory level when there is enough available memory and not much unfinished workload, the SSM may not be detecting correct memory usage. This can cause the boundary event not to be triggered properly. If this situation occurs, try setting the following environment variables for the SSM in your application profile:

- <env name="MMAP_THRESHOLD">131072</env>
- <env name="MMAP_MAX">65536</env>

Note:

The MMAP_THRESHOLD value should be smaller than the average task input/output size.

Optimizing data paging for non-recoverable sessions

To speed up paging and session manager recovery, the specified directory can be on the local drive since the paged data of non-recoverable sessions does not need to be persisted at a shared location.

The following elements and attributes in the SOAM section of the application profile are related to task message data and common data paging for non-recoverable sessions:

- SOAM > PagingTasksInputNonRec > path
- SOAM > PagingTasksOutputNonRec > path
- SOAM > PagingCommonDataNonRec > path
- SOAM > PagingCommonDataUpdatesNonRec > path

For detailed descriptions of these attributes, see the *Platform Symphony Reference*.

Change the session scheduling interval

The frequency at which the scheduler reassesses resource assignments between sessions is called the scheduling interval.

1. Change the interval in the application profile, by setting the `sessionSchedulingInterval` attribute in the Consumer section. The attribute is in milliseconds, default 500 milliseconds.

   For example:

```
<Consumer applicationName="MyApplication" consumerId="/consumer" taskHighWaterMark="1.0"
taskLowWaterMark="0.0" preStartApplication="false" numOfSlotsForPreloadedServices="1"
sessionSchedulingInterval="500" />
```

2. Re-register your application with the `soamreg` command. (If you prefer, do this procedure using the Platform Management Console to export and import the application profile.)

How resources are scheduled by the session manager

Platform Symphony supports the following session scheduling policies:

- Proportional scheduling

  Proportional scheduling allows each session to make some progress, i.e., each session is assigned a number of resources based on its relative priority to other sessions. As the number of pending tasks for a session decreases, the surplus resources are distributed proportionally to deserving sessions.
- Minimum services

  Minimum services ensure that service instances that have loaded and processed the initial data (common data) from a particular session maintains affinity to the session; these service instances can then be used for other similar tasks rather than sending them randomly to a different service instance and reloading the data each time. Refer to *Maintaining data affinity between a session and service instances* on page 413
- Priority scheduling

  Priority scheduling ensures that high-priority sessions with time-critical tasks receive as many available resources as they can use in order to finish as soon as possible. Tasks from lower priority sessions may be preempted. Refer to *Prioritizing sessions for time-critical workload* on page 450

For more information about configuring session scheduling policies in the application profile, refer to the *policy* topic in the *Platform Symphony Reference*.

Prioritizing sessions for time-critical workload

The session manager provides a priority scheduling policy where all resources are assigned to the highest priority session before lower priority sessions. If the highest priority session cannot make use of all of the resources, any unused resources are assigned to the next highest priority session. This cycle repeats to the next highest priority session, and so on. If there are still some slots left after all the sessions are satisfied, these slots are not assigned to any session.

If two sessions have the same priority, the session that is created earlier is treated as though it has higher priority.

Note:

A client application can update the session priority via a Symphony API after the session is created. The priority can also be updated by an administrator using the CLI or PMC.

Tasks within a session are dispatched in first come, first served (FCFS) order.

The following example demonstrates the behavior of the Priority Scheduling policy for sessions with different priorities. The higher priority session gets resources before lower priority sessions.

Preconditions:

- The cluster only has one slot.
- t1: a session starts with default session type, priority set to 1, and 100000 short tasks (one second task runtime).
- t2: a second session starts with default session type, priority set to 10, and 10 long tasks (10 seconds task runtime) while the first session is still running.
- t3: a third session starts with default session type, priority set to 100, and one long task (10 seconds tasks runtime) while the second session is still running.

Post conditions:

1. When the first session is created and has outstanding workload at t1, the one resource is assigned to the first session. The first session continues to use it to run its tasks until a higher priority session requires the resource.
2. When the second session is created and has outstanding workload at t2, the resource is assigned to the second session (since it has higher priority) after the current running task from the first session is done.
3. When the third session is created and has outstanding workload at t3, the resource is assigned to the third session (since it has higher priority) after the current running task from the second session is done.
4. The third session uses the slot to finish all of its tasks. Afterward, the second session receives the slot, since it now has the highest priority, and continues to use it until it finishes all of its tasks. Finally, the first session receives the slot and uses it to finish all of its tasks.

Preemption

If preemption is configured, Symphony takes back resources from lower priority sessions and gives them to higher priority sessions by terminating the currently running tasks of the lower priority sessions. If preemption is not configured, the highest priority session receives resources only after the currently running tasks finish in lower priority sessions.

You can specify session preemption via the PMC when you configure the application. Alternatively, a client application can specify session preemption via a Symphony API when the session is created. Session preemption cannot be updated once the session is created.

You can specify the preemptive option for each SessionType in the application profile. If not specified, the default is the session is not preemptive.

- Any high priority session that is configured as preemptive preempts lower priority session(s) (from lowest to highest priority) if the high priority session has any unsatisfied workload. Note that the service instances assigned to lower priority sessions are terminated and restarted before they can be assigned to the next session.
- Resources released as a result of preemption are distributed to the highest priority session before lower priority sessions. This could mean that the session that triggers preemption may not get the resources in the case that there are even higher priority sessions with unsatisfied workload.

The following example demonstrates the behavior of the R_PriorityScheduling policy when there is a mixture of preemptive and non-preemptive sessions with different priorities in the system.

Preconditions:

- The cluster has five slots.
- Sessions:

Session	Priority	Preemptive flag
First session	100	false
Second session	10	true
Third session	1	false

- The application is already running.
- There is no current workload for the first and second sessions. At this point, the third session has all five resources since the higher priority sessions do not need them.

The client application submits one task to the first session and four tasks to the second session before the next scheduling cycle.

Post conditions:

1. At the next scheduling cycle:
    - The policy will not preempt on behalf of the first session, since the first session is configured as non-preemptive.
    - The policy will preempt four running tasks from the third session, to satisfy the second session's four pending tasks.
2. After preemption occurs, one resource will go to the first session and three resources will go to the second session.

Note:

Even though the first session was not the trigger for preemption, it will be assigned the resource first, according to policy. It will benefit from the second session's preemptive behavior.

Although the second session is not the recipient of those resources, the preemptive behavior is still helping the second session to get ahead. It allows higher priority sessions to get resources faster, which means that the higher priority sessions will finish using the resources faster, and the preemptive session's turn to receive those resources will come faster as well.

3. At the next scheduling cycle, the policy will preempt the last running task from the third session to satisfy the second session, which still has unmet demand.
4. After preemption occurs, the last resource will go to the second session.
5. The third session will receive resources when the first and second session no longer need them.

Configure your application profile for priority scheduling

With this scheduling policy, you assign a priority to a session type relative to the priority of other sessions types; this establishes a rank of importance among the session types. The priority scheduling policy is configured in the application profile.

Note:

If you are editing the application profile outside the Platform Management Console, in the Consumer section, add the parameter `policy="R_PriorityScheduling"`. In the session types section, add the parameter `priority` and register the application with the `soamreg` command.

1. In the Platform Management Console, click Symphony Workload > Configure Applications.

   The Applications page displays.
2. Select the application you want to modify.

   The Application profile page displays.
3. Select SSM scheduling policy to expand it, then under Policy Name, select R_PriorityScheduling.
4. Select Session Type Definition to expand it.
5. Click Add to add a new session type or modify an existing session type.
6. Define the Priority.
7. Choose whether sessions of this type are preemptive, or not.
8. Add new session type definitions or modify existing ones with different priorities, if necessary.
9. Click Save to apply your changes.

Filtering resource groups for sessions

When resource groups are specified at the application level, it means that any sessions serving the application will be allocated resources from the specified resource groups. Sometimes sessions for an application may want to use different resource groups. For example, with a parent/child workload pattern in one application, the parent sessions may only want to use the resource group for which it has 100% ownership, while child session can share resources with other applications. The resource group filter can serve this purpose. Each session can use a resource group filter to narrow down the choices of resource groups specified at the application level.

A resource group filter is a list of resource groups from which the session can use resources. The default value is empty, which means resource groups specified at the application level will be used. Each session has a resource group filter that can narrow down the choices to specific types of resources.

When clients submit workload for sessions with a specified resource group filter, the SSM makes an allocation request to EGO based on the specified requirement. EGO goes through all the resource groups in the resource group filter and tries to satisfy the demand. (The order in which EGO goes through the resource groups is defined by the order that the resource groups appear in the resource plan in the ConsumerTrees.xml file.) After receiving resources from EGO, the SSM assigns resources to its sessions according to the resource group filter for each session.

Configuring resource group filtering for sessions

Resource group filtering at the session level is configured in the SessionTypes element of the application profile using the resourceGroupFilter attribute. The value of resourceGroupFilter must match or be a subset of the resource group name specified at the application level.

Example:

```
<SessionTypes>
<Type name="typeA" ... resourceGroupFilter="RG1,RG2"/>
</SessionTypes>
```

The resourceGroupFilter attribute can be configured through the PMC or by manually editing the application profile.

Overriding configured parameters via API

At session creation time, a client can override the session type’s resourceGroupFilter via the API. Once the session is created, the parameter cannot be changed. Refer to the API Reference in the Knowledge Center for more information.

Specify criteria for resource selection

When you put your application on the grid in Symphony, you specify a resource requirement string in the application profile to set criteria for selecting a resource or restricting which resources are available to the application.

Note that if you set a resource requirement string, and no hosts match your criteria, no hosts will be available for your application.

Note:

(Not applicable to Symphony DE) To find out the ostype to put into the resource requirement string, use the command egosh resource view with the host name.

---

1. For example, if you have heterogeneous machines and your service can only run on one type of machine, such as Windows or Linux, set the resource requirement in the `Consumer` section

   For example:

```
 <Consumer applicationName="MyApplication" consumerId="/consumer" taskHighWaterMark="1.0"
taskLowWaterMark="0.0" preStartApplication="false" numOfSlotsForPreloadedServices="1"
resReq="select(NTX86)" resourceGroupName="ComputeHosts"/>
```

2. Re-register the application with the `soamreg` command. (If you prefer, you can use the Platform Management Console to export and import the application profile.)

Control when applications request or release resources through high- and low-water marks

You can tune your resource requests by specifying a low-water mark in the application profile. The high-water mark is a fixed value.

Both high-water mark and low-water mark are expressed as the ratio of the number of unprocessed tasks to the number of service instances. Unprocessed tasks include both running and pending tasks.

Together, the high-water mark and low-water mark define a range of satisfactory slot allocation, in which the application does not need to request additional resources or release excess resources; refer to the Reference guide for more information about watermark configuration.

- High-water mark

  High-water mark defines the threshold for the application as a whole, to request more resources in order to meet its service level requirement. It defines a ratio of unprocessed tasks of open sessions to service instances. The value of the high-water mark is fixed at 1. The SSM requests enough resources to satisfy this demand.

  For example, a session with a service-to-slot ratio of 1:1, requests at least one CPU slot for every unprocessed task. If the service-to-slot ratio is set to 1:4, i.e., a task requires 4 slots to run, at least 4 slots for every unprocessed task is requested.
- Low-water mark allows you to define the threshold for the application as a whole, to return resources that are no longer needed.

  Once the ratio of unprocessed tasks to service instances falls below the taskLowWaterMark, resources are released and made available for other applications to use.

  The following table summarizes the effects of taskLowWaterMark settings.

Setting	Result
0	SSM does not release any slots
1	SSM releases idle slots when there is no pending workload

Here is an example that illustrates the behavior of taskLowWaterMark for the specified conditions.

taskLowWaterMark: 1

serviceToSlotRatio: 1:1

numUnfinishedTasks: 100

This means that with 100 unprocessed tasks and a taskLowWaterMark of 1, the SSM will keep 100 slots.

If the serviceToSlotRatio is set to 1:4, i.e., each service requires 4 slots, the SSM will keep 400 slots.

1. The low-water mark is configured in the application profile, Consumer section.

   For example:

```
<Consumer applicationName="MyApplication" consumerId="/consumer" taskHighWaterMark="1.0"
taskLowWaterMark="0.0" preStartApplication="false" numOfSlotsForPreloadedServices="1"/>
```

2. Re-register the application with the `soamreg` command. (If you prefer, you may perform these steps using the Platform Management Console to export and import the application profile.)

Deployment performance tuning

You must be a cluster administrator to perform this task.

Repository Server scalability allow you to resolve performance issues in the service deployment. To tune performance, configure the rs.xml file.

- The RS_MAX_DOWNLOAD parameter optimizes bandwidth usage and prioritizes network traffic according to the configured value. This means that the RS limits the number of concurrent downloads to the compute hosts in the cluster to the configured value. The default value is 10.
- The RS_DOWNLOAD_CHUNK_SIZE_KB parameter defines the chunk size in kilobytes. This parameter enables soamdeploy to resume package transfer from a previous break point if the connection is broken for any reason. The default value is 1024. The valid range is 64-10024 kilobytes.

1. Log on to the master host in the cluster.
2. Open the `rs.xml` configuration file, located in the `eservice` directory under the directory in which Symphony was installed.

   For example, on Windows

   %EGO_CONFDIR%\..\..\eservice\esc\conf\services\rs.xml

   For example, on Linux/UNIX

   $EGO_CONFDIR/../../eservice/esc/conf/services/rs.xml
3. Configure the number of active downloads and chunk size in `<ego:ActivitySpecification>`.

```
<sc:ServiceDefinition xmlns:sc="http://www.platform.com/ego/2005/05/schema/sc" xmlns:ego="http://
www.pla/www.platform.com/ego/2005/05/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://
www.platform.com/ego/2005/05/schema/sc ../sc.xsd http://www.platform.com/ego/2005/05/schema ../
ego.xsd" ServiceName="RS">
 ...
<sc:ActivityDescription>
<ego:Attribute name="hostType" type="xsd:string">NTX86</ego:Attribute>
 <ego:ActivitySpecification>
 ...
  <ego:EnvironmentVariable name="RS_MAX_DOWNLOAD">10</ego:EnvironmentVariable>
  <ego:EnvironmentVariable name="RS_DOWNLOAD_CHUNK_SIZE_KB">1024</ego:EnvironmentVariable>
 </ego:ActivitySpecification>
</sc:ActivityDescription>
...
```

Note:

The default size of the RS_DOWNLOAD_CHUNK_SIZE_KB element is 1 megabyte.

4. Save the file.
5. Log on to the master host.
   a) Restart EGO on the master host.

      egosh ego restart

   b) Stop the services:

      egosh service stop RS

   c) Start the services:

      egosh service start RS

PART

V

Fault Tolerance and Failover

CHAPTER

22

Symphony Fault Tolerance

Feature: Automatic failure recovery

The automatic failure recovery feature ensures maximum resource availability to run your workload when a system component fails or becomes unavailable due to a power outage, network failure, application deficiency, or other cause.

This feature is not applicable in Symphony DE.

About automatic failure recovery

Purpose of automatic failure recovery

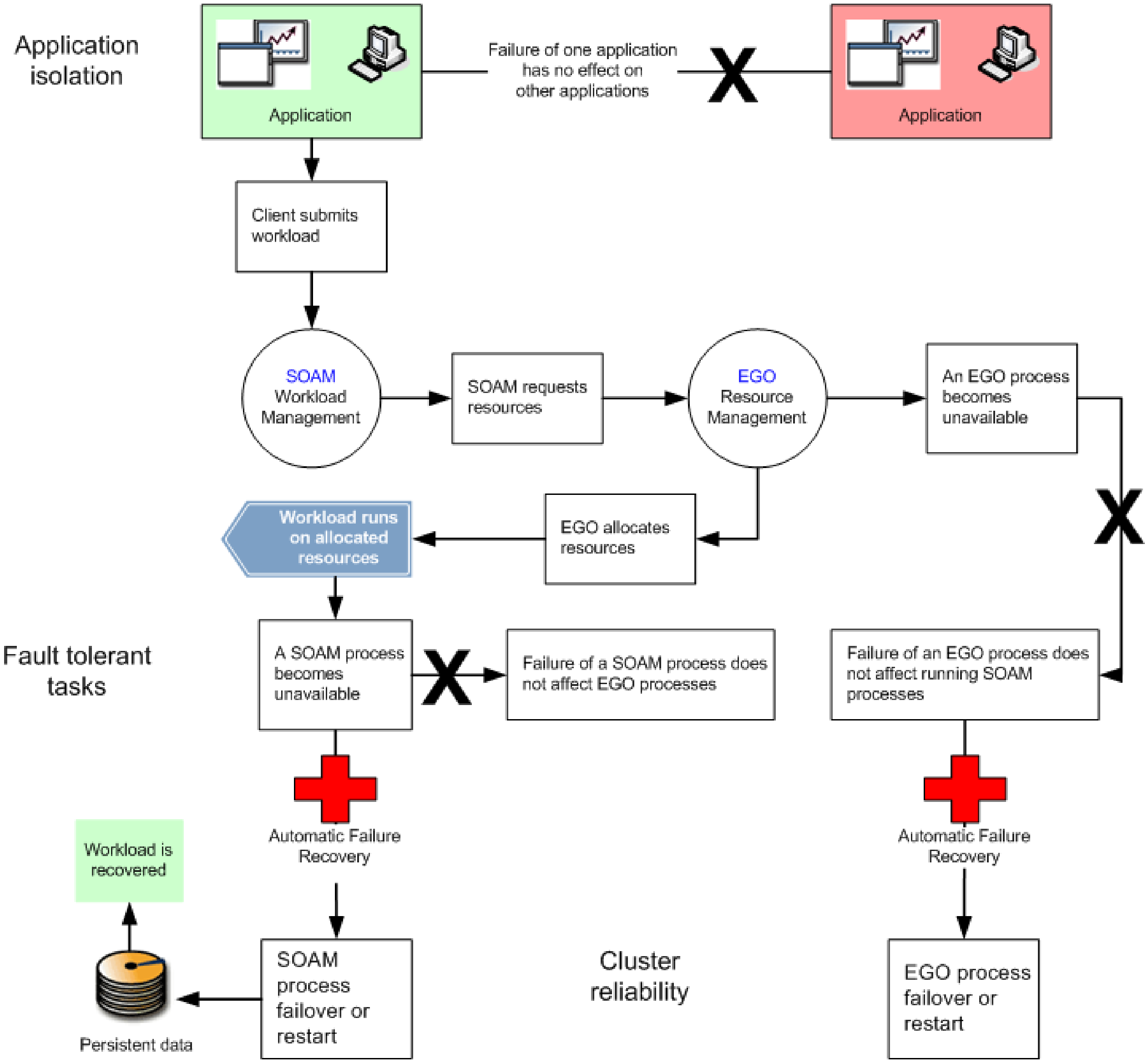
Automatic failure recovery provides a way for the system to automatically restart critical system services and enables you to customize application (service) error handling for each of your applications. Symphony handles a number of failure recovery scenarios.

Benefits of automatic failure recovery

Automatic failure recovery provides a number of benefits, including:

- Application isolation—failure of one application does not affect any other applications, and failure or unavailability of a resource management (EGO) component has no impact on running workload.
- Fault tolerant tasks—with recoverable workload configured, automated failover and data persistence ensures that running workload submitted by an application client continues to run without user intervention when system processes or hosts fail.
- Cluster reliability—master host failover and automatic restart of critical system services ensures high resource availability.

The following illustration shows the benefits of the automatic failure recovery feature once all workload management (SOAM) and resource management (EGO) components have started successfully. In this example, the application profile defines a recoverable session (workload) and the cluster administrator has defined a list of master candidates.


Application isolation
Application
Failure of one application has no effect on other applications
X
Application
Client submits workload
SOAM Workload Management
SOAM requests resources
EGO Resource Management
An EGO process becomes unavailable
X
Workload runs on allocated resources
EGO allocates resources
Fault tolerant tasks
A SOAM process becomes unavailable
X
Failure of a SOAM process does not affect EGO processes
Failure of an EGO process does not affect running SOAM processes
Automatic Failure Recovery
Automatic Failure Recovery
Workload is recovered
SOAM process failover or restart
Cluster reliability
EGO process failover or restart
Persistent data

Scope

Applicability	Details
Operating system	- All host types supported by the Symphony system.
Dependencies	- For master host failover, you must specify one or more master host candidates. - Files required for failover must be on a shared file system. - Cluster administrator and consumer user accounts must have operating system permissions to access directories on the shared file system.
Limitation	- Symphony does not provide automatic failure recovery of the shared file system if the shared file system becomes unavailable.

Configuration to enable automatic failure recovery

Automatic failure recovery is enabled for automatic process restart for critical system services and for restart of Symphony workload management (SOAM) components. Automatic failure recovery for applications is enabled by default in the application profile. You can also enable

- Session manager failover
- Session recovery, which makes workload recoverable
- Master host failover

Configuration to enable session manager failover

Session manager failover is enabled by default when you use a shared file system and do not change the default values for any of the following attributes:

- SOAM > SSM > resourceGroupName
- SOAM > SSM > workDir
- SOAM > DataHistory > path
- SOAM > PagingTasksInput > path
- SOAM > PagingTasksOutput > path
- SOAM > PagingCommonData > path
- SOAM > PagingCommonDataUpdates > path
- SOAM > JournalingTasks > path
- SOAM > JournalingSessions > path
- SOAM > JournalingSessionTagConfig > path

Changing any of these attributes could affect session manager failover. For detailed descriptions of these attributes, see the *Platform Symphony Reference*.

Configuration to enable session recovery

Defining a recoverable session makes workload recoverable after session manager failover or restart.

Section	Attribute name and syntax	Behavior
SessionTypes > Type	recoverable=true | false	- Specifies whether the session can be recovered after session manager failover or restart. If true, Symphony persists the common data and its update (if any) for the session, task data for tasks that have not yet been returned to the client, and data required to reconstruct those objects. If false (default), the system does not persist session and task data, and tasks must be rerun. **Important:** If the file system that is used by the SSM for paging and recovery purposes is not stable, you need to configure flushDataAsap=”true” in the application profile. This causes the SSM to write data to disk directly, rather than using system cache, before continuing with the next operation. This guarantees the data is actually stored on the disk and not in system cache after the SSM finishes the disk write operation. The SSM will be able to read the data back in case of recovery. Refer to the application profile reference for details about this parameter. It is strongly recommended to use a stable and reliable file system in your Symphony cluster to avoid losing any data.

Configuration to enable master host failover

The master candidate list defines which hosts are master candidates. By default, the list includes just one host, the master host, and there is no failover. If you configure additional candidates to enable failover, the master host is first in the list. If the master host becomes unavailable, the next host becomes the master, and so on down the list.

For master candidate failover to work properly, the master candidates must share a file system that must always be available.

Important:

The shared file system should not reside on a master host or any of the master candidates. If the shared file system resides on a master host and the master host fails, the next candidate cannot access the necessary files.

If you have configured at least one management host for your cluster in addition to the master host but have not selected any failover candidates, the Platform Management Console dashboard displays a reminder message in red with a link to the page from which you define the master candidate list.

Configuration source	Setting	Behavior
Platform Management Console: **Cluster > Summary >Master Candidates**	• **Add** available hosts to the **Master Candidates** list, or **Remove** hosts from the **Master Candidates** list. • Rearrange the order of master candidates: *host_name* > **Up** | **Down**	• The master candidates are now set in the order you want them to fail over. The cluster automatically restarts when you click **Apply**, making the changes take effect. • All master candidates must be selected from the available management hosts. A compute host cannot be a master candidate. • The default configuration of the EGOManagementServices consumer provides for master candidate failover; do not change the number of slots owned by this consumer.

Alternatively, you can use the command line interface to specify a list of master candidates.

Command	Description
`egoconfig 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. Do not specify compute hosts in this list. **Caution:** Include all master candidates in the list when you issue this command; `egoconfig masterlist` overwrites the existing list.

Automatic failure recovery behavior

Automatic failure recovery behavior depends on which process fails or becomes unavailable, and on the type of host on which the process runs.

Recovery when individual processes fail or become unavailable

The following description provides details about what happens when a workload management (SOAM), Platform Management Console, reporting, or resource management (EGO) process fails or becomes unavailable independently of other processes.

Important:

Recovery of any workload management (SOAM), Platform Management Console, Reporting, or resource management (EGO) process usually takes less than one or two minutes, and can take as little as one or two seconds, provided that the host remains available.

When this process is in failure recovery…	The effects are…		
	Workload	**Resource allocation**	**Lifecycle or other processes**
Workload management (SOAM) processes			
Service instance (si)	You can define the actions retry or fail for the SessionEnter, SessionUpdate, and Invoke methods.	If blockHost is defined as the actionOnSI for a service instance exit, timeout, exception, or control code, the system terminates the running service instance on this host and does not use this host to start any other service instance for the application. If restart is defined as the actionOnSI, the service instance tries to restart on the original host.	You can define the following actions for service instances based on specific states of the service lifecycle: keepAlive, restartService or blockHost. The session manager will continue to run the service, restart the service on the same host, or—through communications with the Virtual Execution Machine Kernel Daemon (vemkd)—block the host for use by the application associated with the service.
Service instance manager (sim)	The session manager requeues and reruns tasks for the session that was running on the service instance manager; no workload is lost.	If blockHostOnTimeout= "true" in the SOAM > SIM section of the application profile and if, after a service instance manager is started, the service instance manager process cannot contact the session manager within the startUpTimeout, the system does not use this host to start any other service instance managers for the application. If blockHostOnTimeout= "false", the system tries again to start the service instance manager on the original host.	If the service instance manager dies after starting successfully, the associated service instance exits. The session manager then restarts the service instance manager.

When this process is in failure recovery…	The effects are…		
	Workload	**Resource allocation**	**Lifecycle or other processes**
Session manager (ssm)	For recoverable sessions, the session manager persists the information needed to resume the workload without loss of data, and session manager failover or recovery is transparent to the client application. For non-recoverable sessions, the workload is lost and the client must resubmit the workload.	When it restarts, the session manager re-registers with the resource management component (EGO) and obtains a list of resources that were previously allocated to the session manager. The session manager stops and restarts all running service instance managers on those resources.	The service instance managers associated with the failed session manager also die, and requests from the Platform Management Console and command line interface fail. The session director restarts the session manager. On restart, the session manager reads only the task and session control objects, not the input/output messages; the session manager reads those messages as required when dispatching a task. Session manager monitoring information resets; the following statistical values apply to the time period that begins with session manager restart. • Closed sessions since SSM started • Aborted sessions since SSM started • Time of the last session aborted • Done tasks since SSM started • Error tasks since SSM started • Time of the last error task When the session manager is unavailable, clients cannot create new SDK connections. • If the client is already connected and the session manager becomes unavailable, the Symphony APIs retry the connection. • If the client has not yet connected and the session manager is unavailable, the client receives an exception and must wait for the session manager to become available.

When this process is in failure recovery…	The effects are…		
	Workload	**Resource allocation**	**Lifecycle or other processes**
Session director (sd)	Session director failure has no impact on running workload; the session manager handles workload execution. For new workload, clients submitting workload wait momentarily for the EGO service controller to restart the session director.	Session director failure has no impact on resource allocation. The session director saves information about the resources it uses and, after restart, uses the same resources.	While the session director is down momentarily, requests from the Platform Management Console and command line interface fail. If you set view preferences for the dashboard to automatically refresh, the request succeeds once the session director has restarted. When the session director is unavailable, clients cannot create new SDK connections. • If the client is already connected and the session director becomes unavailable, the Symphony APIs retry the connection. • If the client has not yet connected and the session director is unavailable, the client receives an exception and must wait for the session director to become available. The EGO service controller usually restarts the session director within a few seconds on the original host or on a new host if the original host has no available resources. The EGO service controller tries up to 10 times to restart the session director before setting the status to ERROR.

When this process is in failure recovery…	The effects are…		
	Workload	**Resource allocation**	**Lifecycle or other processes**
Repository service (rs)	Repository service failure has no effect on running workload. New workload that needs to download a service package must wait until the repository service becomes available.	Repository service failure has no effect on resource allocation.	The EGO service controller restarts the repository service on the original host or on a new host if the original host has no available resources. The EGO service controller tries up to 10 times to restart the repository service before setting the status to ERROR.
Platform Management Console processes			
Web service manager (wsm)	Web service manager failure has no effect on workload.	Web service manager failure has no effect on resource allocation.	The EGO service controller restarts the Web service manager on the original host or on a new host if the original host has no available resources. The EGO service controller tries up to 10 times to restart the Web service manager before setting the status to ERROR. The web service manager monitors the java process of TOMCAT—a key component of the Platform Management Console—and restarts the java process if it goes down.
Reporting processes			

When this process is in failure recovery…	The effects are…		
	Workload	**Resource allocation**	**Lifecycle or other processes**
Platform loader controller (plc)	Loader controller failure has no effect on workload.	Loader controller failure has no effect on resource allocation.	If the loader controller becomes unavailable, the Platform Enterprise Reporting Framework cannot collect sampling data for reporting purposes. The EGO service controller restarts the loader controller on the original host or on a new host if the original host has no available resources.The EGO service controller tries up to 10 times to restart the loader controller before setting the status to ERROR.
Data purger (purger)	Data purger failure has no effect on workload.	Data purger failure has no effect on resource allocation.	If the data purger becomes unavailable, the database could temporarily grow until the data purger recovers and can once again purge the data. The time it takes for the database to run out of space depends on the size of your system. The EGO service controller restarts the data purger on the original host or on a new host if the original host has no available resources.The EGO service controller tries up to 10 times to restart the data purger before setting the status to ERROR.
Resource management (EGO) processes			
Master load information manager (master lim)	Master load information manager failure has no effect on running workload. Clients submitting new workload receive an exception.	The system considers the master host unavailable and a master candidate takes over as master host. During failover to the master candidate, the system does not respond to resource allocation requests.	If no master candidate is available, the cluster is down. The system cannot restart the master load information manager; you can manually restart it, however, using the `egosh ego start all` command.

When this process is in failure recovery…	**The effects are…**		
	Workload	**Resource allocation**	**Lifecycle or other processes**
Virtual Execution Machine Kernel Daemon (vemkd)	Virtual Execution Machine Kernel Daemon failure has no effect on running workload. Clients submitting new workload receive an exception.	During failure recovery, the system does not respond to resource allocation requests.	The master load information manager restarts the Virtual Execution Machine Kernel Daemon.
Process execution monitor (pem)	Process execution monitor failure has no effect on running workload.	Process execution monitor failure has no effect on resource allocation.	The load information manager restarts the process execution monitor on a compute or management host. The master load information manager restarts the process execution monitor on the master host.
EGO service controller (egosc)	EGO service controller failure has no effect on running workload.	EGO service controller failure has no effect on resource allocation.	The Virtual Execution Machine Kernel Daemon restarts the EGO service controller.
Load information manager (lim)	The system considers the host unavailable and terminates workload on the unavailable host. EGO notifies the SOAM component (session director or session manager) that has been allocated to the unavailable host. The session director or session manager stops the service (service instance and service instance manager) on that host and requests another resource.	The system does not allocate any resources on the unavailable host.	The master load information manager restarts the load information manager on the compute or management host.

Recovery when hosts fail

When processes become unavailable in combination because of a hardware failure, you see the following behavior.

Note:

The majority of the time required for failover of compute, management, and master hosts is used to confirm that the host is actually unavailable.

This prevents temporary network delays or instability from triggering frequent and unnecessary host switches.

When this host is down…	The effects are…
Compute host	• The following processes become unavailable during failure recovery: • Load information manager • Process execution monitor • Service instance manager • Service instance • When the session manager-service instance manager connection breaks, the session manager requeues the affected tasks. If the session manager does not recognize the broken connection, the resource manager (EGO) notifies the session manager within three minutes that the host is down. • The session manager requests a new resource. • Workload runs on the new compute host.
Management host	• The following processes become unavailable during failure recovery: • Load information manager • Process execution monitor • Session director • Session manager • Repository service • Web service manager • Loader controller • Data purger • In less than three minutes, a new management host takes over and gets configuration information from the shared configuration directory.

When this host is down…	The effects are…
Master host	• The following processes become unavailable during failure recovery: • Master load information manager • Virtual Execution Machine Kernel Daemon • Process execution monitor • EGO service controller • Session director • Repository service **Note:** The session director and repository service can run on any management host; they will become unavailable during failure recovery only if they are running on the master host. • By default, in less than two minutes, a management host from the master candidates list takes over and gets configuration information from the configuration directory on the shared file system. When the primary master host recovers, it takes over from the master candidate. The load information manager on the primary master becomes the master load information manager, and the Virtual Execution Machine Kernel Daemon and EGO service controller processes on the master candidate host are terminated and restarted on the primary master host. All other EGO services, including SOAM processes remain running on their current host.

Configuration to modify automatic failure recovery

You can modify

- Automatic failure recovery behavior for an application
- Service instance error handling—actions for unexpected exits, timeouts, exceptions, or control codes
- Actions for a timeout between the service instance manager and the session manager

Configuration to modify automatic failure recovery for an application

The following attributes and environment variables can be configured to change the way that automatic failure recovery works once it is enabled for an application.

Configuration source	Setting	Behavior
Application profile: Consumer	flushDataAsap=true | false	• 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 (default), data is cached before it is written to disk. **Important:** Setting this parameter to true could substantially degrade performance.
	transientDisconnectionTimeout= *seconds*	• Specifies the number of seconds the session manager waits for the client to reconnect before it aborts the session when the connection between the client and session manager is broken. • Specify an integer equal to or greater than 1. The default value is 30 seconds. • 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.
	ioRetryDelay=*seconds*	• Specifies the number of seconds to wait before retrying an I/O operation after a previous failure. • Specify an integer equal to or greater than 1. The default value is 1.
Application profile: SOAM > SSM	resReq=**"select(***select_string***)" "select (***select_string*) **order(***order_string***)"**	• Describes the criteria for defining a set of resources to run session managers. Session managers should run on management hosts. When specifying a resource requirement string, you must indicate the select string `"select(mg)"` so that only management hosts are selected to run session managers. • The default value is "", which specifies any host in the ManagementHosts resource group.

Configuration source	Setting	Behavior
Application profile: SessionTypes > Type	abortSessionIfClientDisconnect=true | false	• Specifies whether the session is aborted if the session manager detects that the connection between the client and the session manager is broken. The default value is true. • Used with the transientDisconnectionTimeout attribute.

Configuration to modify service instance error handling behavior

Section	Method	Attribute name and syntax	Behavior
Service > Control > Method > Timeout	• Register • CreateService • SessionEnter • SessionUpdate	actionOnSI=restartService|blockHost	• Specifies whether to restart the service or block the host on timeout. • The default for Register, CreateService, and SessionEnter, SessionUpdate is blockHost.
	• Invoke • SessionLeave		• The default for Invoke and SessionLeave is restartService.
	• SessionEnter • SessionUpdate • Invoke	actionOnWorkload=retry | fail	• Specifies whether to retry the method (default) up to the number of times configured by the session and task retry limits or abort the session (SessionEnter or SessionUpdate)/fail the task (Invoke). **Note:** The retry count for both SessionEnter and SessionUpdate methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3.

Section	Method	Attribute name and syntax	Behavior
Service > Control > Method > Exception	• CreateService	actionOnSI=restartService| blockHost	• Specifies whether to restart the service or block the host (default) when the specified exception (failure or fatal exception) occurs.
	• Invoke • SessionEnter • SessionUpdate • SessionLeave	actionOnSI=keepAlive | restartService | blockHost	• Specifies whether to continue running the service (default), restart the service, or block the host when the specified exception (failure or fatal exception) occurs.
	• SessionEnter • SessionUpdate • Invoke	actionOnWorkload=retry | fail	• Specifies whether to retry the method up to the number of times configured by the session and task retry limits or abort the session (SessionEnter or SessionUpdate)/fail the task (Invoke). **Note:** The retry count for both SessionEnter and SessionUpdate methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3.

Section	Method	Attribute name and syntax	Behavior
Service > Control > Method > Exit	• Register • CreateService • SessionEnter • SessionUpdate	actionOnSI=restartService|blockHost	• Specifies whether to restart the service or block the host on if the service process exits during the execution of the method. • The default for Register, CreateService, and SessionEnter, SessionUpdate, is blockHost.
	• Invoke • SessionLeave		• The default for Invoke and SessionLeave is restartService.
	• SessionEnter • SessionUpdate • Invoke	actionOnWorkload=retry | fail	• Specifies whether to retry the method (default) up to the number of times configured by the session and task retry limits or abort the session (SessionEnter or SessionUpdate)/fail the task (Invoke). **Note:** The retry count for both SessionEnter and SessionUpdate methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3.

Section	Method	Attribute name and syntax	Behavior
Service > Control > Method > Return	• CreateService • SessionEnter • SessionUpdate • Invoke • SessionLeave	actionOnSI=keepAlive | restartService | blockHost	• Specifies whether to continue running the service (default), restart the service, or block the host when the method returns normally and specified code is returned.
	• SessionEnter • SessionUpdate • Invoke	actionOnWorkload=retry | fail | succeed	• Specifies whether to consider the method task as having reached completion based on a normal return (default), retry the method up to the number of times configured by the session and task retry limits, or abort the session (SessionEnter or SessionUpdate)/fail the task (Invoke). **Note:** The retry count for both SessionEnter and SessionUpdate methods are considered together. For example, if SessionEnter fails once and SessionUpdate fails twice, then the session rerun count is equal to 3.

Section	Method	Attribute name and syntax	Behavior
SessionTypes > Type	• Invoke	taskRetryLimit=*integer*	• Specifies the number of attempts to retry a task before the system fails the task. • The value can be 0 or greater. If you specify a value of 3 (default), the system makes 1 attempt to run the task followed by 3 retries before the task fails.
	• SessionEnter • SessionUpdate	sessionRetryLimit=*integer*	• Specifies the number of times the session can retry binding to the service before the session is aborted. • The value can be 0 or greater. If you specify a value of 3 (default), the system makes 1 initial attempt to run the SessionEnter or SessionUpdate methods followed by 3 retries before the system aborts the session.

Configuration to modify service instance manager-session manager timeout actions

You can change how the system handles a timeout between the service instance manager and the session manager.

Section	Attribute name and syntax	Behavior
SOAM > SIM	blockHostOnTimeout="true" | "false"	• If "true" (default), blocks the host for the application when the service instance manager times out while trying to communicate with the session manager. This means that the services associated with the application run on a different host than the one on which the timeout occurred. If "false", the service tries to restart on the original host. • Used with the startUpTimeout attribute.
	startUpTimeout="*seconds*"	• Number of seconds to wait for the service instance manager to communicate with the session manager. The default is 60 seconds. This parameter works in conjunction with blockHostOnTimeout. • After a service instance manager is started, if the service instance manager cannot contact the session manager within the startUpTimeout and if blockHostOnTimeout="true", the session manager requests a new host from EGO and tries to start a new service instance manager on the new host.

Automatic failure recovery interface

Actions to submit workload

No actions required. For recoverable sessions, session manager failover or recovery is transparent to the application client.

Actions to monitor

You can monitor automatic failure recovery through the Platform Management Console and from the command line. You can also set up SNMP traps to capture system events.

User	Command	Description
• Cluster administrator	From the Platform Management Console Dashboard	• Displays the overall health and drill-down details of the cluster, services, and workload. When a process restarts, the process ID changes.
• Cluster administrator	From the command line: `egosh resource list -m`	• Displays the list of failover candidate hosts in the cluster and identifies which host is currently the master.
• Cluster administrator	From the SNMP trap notifications: • `SYS_FAILOVER_RETRIED`	• The system is trying to restart the session manager or service instance manager.
	• `SYS_SSM_DOWN`	• The session manager goes down abnormally.
	• `SYS_SSM_UP`	• The session manager comes up.
• Cluster administrator	From the SNMP trap notifications: • `SYS_VEMKD_UP`	• To receive this notification, you must first configure EGO_EVENT_PLUGIN=*plugin_name* and EGO_EVENT_MASK=LOG_INFO in `ego.conf`. • Indicates that the master host has failed over to a new master host, or that the cluster has been reconfigured.

You can also check the progress of failure recovery as follows:

Process	**User**	**Command**	**Description**
• Service instance manager • Service instance	• Cluster administrator • Consumer administrator • Consumer user	From the Platform Management Console Dashboard: **Symphony Workload > Monitor Workload >*application_name***	• The presence of a running task indicates that the service instance manager and service instance processes are available. • If tasks are pending but no tasks are running, the service instance manager and service instance processes might be unavailable.
		From the command line: `soamview app` *app_name* -l	• Displays the number of running and pending tasks for all sessions of an application. The presence of a running task indicates that the service instance manager and service instance processes are available. • If tasks are pending but no tasks are running, the service instance manager and service instance processes might be unavailable.
• Session manager	• Cluster administrator • Consumer administrator • Consumer user	From the command line: `soamview app` *app_name*	• The presence of a session manager process ID indicates that the session manager is available.
• Session director • Repository service • Data purger • Loader controller • Web service manager	• Cluster administrator	From the command line: `egosh service list`	• If the process appears in the `STARTED` state, the process is available.
• Master load information manager • Virtual Execution Machine Kernel Daemon • EGO service controller	• Cluster administrator	From the command line: `egosh service list`	• If the command responds, these processes are available. • If the command does not respond, one of these processes might be unavailable.

Process	User	Command	Description
- Load information manager (non-master) - Process execution monitor	- Cluster administrator	From the command line: `egosh resource list`	- If a host has a status of `ok`, the load information manager and process execution monitor on that host are available.

Actions to control

Not applicable. Automatic failure recovery does not require user intervention.

Actions to display configuration

User	Command	Behavior
- Cluster administrator - Consumer administrator - Consumer user	From the Platform Management Console Dashboard: **Symphony Workload > Monitor Workload > *application_name* > Application Profile**	- Displays settings for all of the application-level automatic failure recovery configuration.
- Cluster administrator - Consumer administrator - Consumer user	From the command line: `soamview app` *app_name* `-p`	- Displays application profile settings for the selected application.
- Cluster administrator	**Cluster > Summary > Master Candidates**	- Displays a list of master candidates and the order in which failover occurs.
- Cluster administrator	From the command line: `egosh resource list -m`	- Displays the list of failover candidate hosts in the cluster and identifies which host is currently the master.

CHAPTER

23

Application Fault Tolerance

Data loss prevention

In a grid environment, there may be hundreds and thousands of compute hosts distributed in a cluster. In a typical risk management application, there may be hundreds of thousands of perturbations of market data/conditions. Each one of these can be a workload unit.

When you submit this workload to a grid, you expect the grid system to distribute the workload on grid, and guarantee processing without losing any workload, even if there are failures in hardware or software in:

- Grid management machines or software
- Compute machines and service applications

A reliable grid system should guarantee a transactional handling of application execution on the grid. A failure or even an entire system reboot should not require rerunning the workload from the beginning.

One problem in a traditional MPI-based parallel application is that when there is a failure in a distributed environment, the MPI-based application may fail and need to rerun from the beginning. Rerunning a large workload or the entire workload in the system not only wastes time and resources, but also may miss the time window of business opportunities.

Add recovery with recoverable sessions

Platform Symphony supports reliable computing by persisting Symphony session and task inputs and outputs. However, sometimes you may not want to recover your workload when a failure or error happens, or, you may want to trade persistency for performance— task persistency takes time and disk space and may slow down the overall system response time.

You can define whether a session is recoverable or non-recoverable in the application profile through the session type. In the client application, you can then specify the appropriate session type in createSession ().

Choose a recoverable session when

- You have a long session that may last hours to compute many CPU-intensive tasks, and you do not want to waste CPU cycles to resubmit tasks in the session if a failure or error occurs.
- It is difficult or impossible to resubmit tasks in the session when a failure or error occurs.
- You have a mission-critical session that has to be finished before a deadline.

Choose a non-recoverable session when

- You have a short session that may only last for minutes, and you can always create a new session to resubmit tasks if a failure or error occurs.
- You want Symphony to immediately clean up the session and release the CPUs if a failure or error happens. Keeping this session running in the system is just waste of CPU cycles.
- You have an interactive online session that requires quick response time.

Implement application-level checkpointing for sessions

If you have long running tasks, you may not want to rerun a task from the beginning in case of failure.

A good practice is to have a long running task that periodically persists its intermediate results, such as every 10 minutes, so that when the task is rerun by Symphony, it can continue from where the last intermediate results that were persisted.

You need a persistent shared location like a persistent shared data cache or a shared file system because a task may be rerun on a different machine than previously.

Once a task can persist its intermediate results, you can perform application-level checkpointing by suspending the session.

A service instance can get an interrupt event by calling `serviceContext.getLastInterruptEvent()`, and use a grace period to persist intermediate results in a persistent shared location. Later on, either when the whole suspended session is resumed, or then the unfinished task is redispatched, another service instance picks up the task, and restores the intermediate results from the shared location.

Configure a recoverable session

A recoverable session can be used to preserve your workload under exceptional circumstances such as a power failure or host failure.

Recoverable sessions can incur additional overhead because the workload must be journaled. Specifying your sessions as recoverable may not be appropriate for all types of workload, since it can take less time to rerun all the tasks in the session rather than to recover and resume them. The time it takes to rerun or recover and resume tasks in a session varies with the data size and number of tasks.

Note:

If you are editing the application profile outside the Platform Management Console, set the parameter `recoverable="true"` in the session types section and re-register the application with the soamreg command.

1. In the Platform Management Console,
2. In the Platform Management Console, click Symphony Workload > Configure Applications.

   The Applications page displays.
3. Select the application to modify.

   The Application Profile page displays.
4. In the Session Type Definition section, in Recoverability, select Recoverable.
5. Click Save to apply your changes.

PART

VI

Troubleshooting Tools

CHAPTER

24

Logs

About EGO log files

Log files contain important run-time information about the general health of EGO daemons and EGO system events. Log files are an essential troubleshooting tool during production and testing.

The naming convention for most EGO log files is the name of the daemon plus the host name the daemon is running on.

The following table outlines the EGO daemons and their associated log file names. Most log files on Windows hosts have a `.txt` extension.

By default, most EGO log files are found in *`EGO_TOP`*`\kernel\log` (Windows) or *`EGO_TOP`*`/kernel/log` (Linux/UNIX).The majority of log entries are informational in nature. It is not uncommon to have a large (and growing) log file and still have a healthy cluster.

Log files can also be accessed through the Platform Management Console (from System Logs > Standard Logs).

List of log files

Daemon or Script	**Component**	**Log file name**	**Windows**	**Linux/UNIX**
	Configuration wizard (add a product to an existing cluster)	`upgradeLog.log`	*`EGO_TOP`*`\gui\logs`	$EGO_TOP/gui/logs
datasourcetools	Reporting: Database Configuration Tool	`datasourcetools.`*`host_name`*`.log`	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
egoconsumeresloader	Reporting: EGO Consumer Resource Data Loader	egoconsumerresloader.*host_name*.log	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
egodynamicresloader	Reporting: Dynamic metric data loader	`egodynamicresloader.host_name.log`	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
egoeventsloader	Reporting: EGO events data loader	egoeventsloader.*host_name*.log	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
egosc	EGO Service Controller	`egoservice.audit.log`, `esc.log.`*`host_name`*	*`EGO_TOP`*`\eservice\esc\log`	`$EGO_TOP/eservice/esc/log`
egostatisticresloader	Reporting: Static attribute data loader	`egostatisticresloader.`*`host_name`*`.log`	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
fam	File Access Manager	• `fam.`*`host_name`*`.log`	*`EGO_TOP`*`\kernel\log`	`$EGO_TOP/kernel/log`

Daemon or Script	**Component**	**Log file name**	**Windows**	**Linux/UNIX**
lim	Load Information Manager (lim)	• `lim.log.`*`host_name`*	*`EGO_TOP`*`\kernel\log`	`$EGO_TOP/kernel/log`
named	Service Director	• `named.log`	*`EGO_TOP`*`\eservice\esd\conf\named\namedb`	`$EGO_TOP/eservice/esd/conf/named/namedb`
pem	Process Execution Manager(pem)	• `pem.log.`*`host_name`*	*`EGO_TOP`*`\kernel\log`	`$EGO_TOP/kernel/log`
pim	Process Information Manager	• `pim.log.`*`host_name`*	N/A (Linux/UNIX only)	`$EGO_TOP/kernel/log`
plc	Loader controller	`plc.`*`host_name`*`.log`	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
purger	Reporting: Data purger	`purger.`*`host_name`*`.log`	*`EGO_TOP`*`\perf\logs`	`$EGO_TOP/perf/logs`
rfa	Remote File Access(rfa): server side	• `cli.`*`host_name`*`.log`	*`EGO_TOP`*`\eservice\rs\log`	`$EGO_TOP/eservice/rs/log`
rfa	Remote File Access: client side	• `cli.`*`host_name`*`.log`	Where `rfa` was run from	Where `rfa` was run from
rs	Repository service (rs)	• `rs.`*`host_name`*`.log` • `repositoryservice.audit.log`	*`EGO_TOP`*`\eservice\rs\log`	`$EGO_TOP/eservice/rs/log`
vemkd	EGO kernel daemon	• `ego.audit.log` • `vemkd.log.`*`host_name`*	*`EGO_TOP`*`\kernel\log`	`$EGO_TOP/kernel/log`
WEBGUI	Platform management console/WEBGUI	• `catalina.out` • `wsm.log.`*`host_name`*	*`EGO_TOP`*`\gui\logs`	`$EGO_TOP/gui/logs`
WSG	Web service gateway	• `wsg.log`	*`EGO_TOP`*`\eservice\wsg\log`	`$EGO_TOP/eservice/wsg/log`

Log entry format

The standard format for log file entries is:

date time_zone log_level [*process_id*:*thread_id*] *action*:*description*/*message*

where the date is expressed in YYYY-MM-DD hh:mm:ss.sss.

For example, `2006-03-14 11:02:44.000 Eastern Standard Time ERROR [2488:1036] vemkdexit: vemkd is halting`.

Why do log files grow so quickly?

Every time an EGO system event occurs, a log file entry is added to a log file. Most entries are informational in nature, except when there is an error condition. If your log levels provide entries for all information (for example, if you have set them to `LOG_DEBUG`), the files grow quickly.

Suggested settings:

- During regular EGO operation, set your log levels to `LOG_WARNING`. With this setting, critical errors are logged but informational entries are not, keeping the log file size to a minimum.
- For troubleshooting purposes, set your log level to `LOG_DEBUG`. Because of the quantity of messages you receive when subscribed to this log level, change the level back to `LOG_WARNING` as soon as you are finished troubleshooting.

Note:

If your log files are too long, you can always rename them for archive purposes. New, fresh log files are then created and log all new events.

How often should I maintain log files?

The growth rate of the log files is dependent on the log level and the complexity of your cluster. If you have a large cluster, daily log file maintenance may be required.

We recommend using a log file rotation utility to do unattended maintenance of your log files. Failure to do timely maintenance could result in a full file system, which hinders system performance and operation.

What can I do if there are no LIM logs generated?

In Windows, you have the option of viewing application and system events using the Event Viewer. To open the Event Viewer:

1. From the Control Panel, select Administrative Tools > Event Viewer.

   The Event Viewer displays.
2. From the list in the left pane, select Application or System.

   The list of events displays.

Log classes and log levels

Configuration files where log level and class information are retrieved

The lim, pem, and vemkd daemons read `ego.conf` to retrieve the following information (as corresponds to the particular daemon).

- `EGO_LOG_MASK`: The log level used to determine the amount of detail logged.
- `EGO_DEBUG_LIM`: The log class setting for lim.
- `EGO_DEBUG_PEM`: The log class setting for pem.
- `EGO_DEBUG_VEMKD`: The log class setting for vemkd.

Fastpath:

- Windows: `%EGO_CONFDIR%\ego.conf`
- Linux/UNIX: *`$EGO_CONFDIR/`*`ego.conf`

The service director daemon (“named”) reads `named.conf` to retrieve the following information:

- logging severity: The configured severity log class controlling the level of event information that is logged (`critical`, `error`, `warning`, `notice`, `info`, `debug`, or `dynamic`). In the case of the log class set to `debug`, a log level is required to determine the amount of detail logged for debugging purposes. The higher the log level number, the more debug details messages are logged. Refer to third-party documentation for more information about BIND and logging.

Fastpath:

- Windows: *`EGO_TOP`*`\eservice\esd\conf\named\conf\named.conf`
- Linux/UNIX: *`EGO_TOP`*`/eservice/esd/conf/named/conf/named.conf`

The egosc daemon reads `egosc_conf.xml`.

Fastpath:

- Windows: *`EGO_TOP`*`\eservice\esc\egosc_conf`.xml
- Linux/UNIX: *`EGO_TOP`*`/eservice/esc/egosc_conf`.xml

The wsg daemon reads `wsg.conf` to retrieve the following information:

- `WSG_LOGDIR`: Where to write `wsg.log` files.

Fastpath:

- Windows: *`%EGO_CONFDIR%`*`\wsg.conf`
- Linux/UNIX: *`$EGO_CONFDIR`*`/wsg.conf`

The wsm daemon reads `wsm.conf` to retrieve the following information:

- `LOG_LEVEL`: The configured log level controlling the level of event information that is logged (`INFO`, `ERROR`, `WARNING`, or `DEBUG`).

Fastpath:

- Windows: *`EGO_TOP`*`\gui\conf\wsm.conf`
- Linux/UNIX: *`EGO_TOP`*`/gui/conf/wsm.conf`

If a system is running well, typically set log level to `info` or even `warning` to minimize messages.

Note:

The daemons associated with the reporting feature read various `.xml` files to retrieve information. For more information, see the Reports chapters.

Log classes for vemkd and pem

Use the following parameters to specify the log class:

- vemkd: `EGO_DEBUG_VEMKD`

  For example, `EGO_DEBUG_VEMKD=LC_AUTH`.
- pem: `EGO_DEBUG_PEM`

  For example, `EGO_DEBUG_PEM=LC_PEM`

Every log entry belongs to a log class. You can use log class as a mechanism to filter log entries by area. Log classes in combination with log levels allow you to troubleshoot using log entries that only address, for example, configuration.

Log classes (as well as log levels) can be filtered at run time using `egosh debug`.

Valid logging classes are as follows:

Log class	Description
LC_TRACE	Logs significant program steps.
LC_COMM	Logs messages related to communications.
LC_AUTH	Logs messages related to users and authentication.
LC_MEM	Logs messages related to memory allocation.
LC_SYS	Logs messages related to system calls.
LC_PERF	Logs messages related to performance.
LC_RSRC	Logs messages related to resources, including host status changes.
LC_ALLOC	Logs messages related to the resource allocation engine.
LC_ACTIVITY	Logs messages related to activities.
LC_PEM	Logs messages related to the process execution manager (pem).
LC_EVENT	Logs messages related to the event notification service.
LC_QUERY	Logs messages related to client queries.
LC_RECOVER	Logs messages related to recovery and data persistence
LC_CONF	Logs messages related to configuration.
LC_CLIENT	Logs messages related to clients.

Log classes for lim

Use `EGO_DEBUG_LIM` to specify the log class. For example, `EGO_DEBUG_LIM=LC_MEMORY`.

Every log entry belongs to a log class. You can use log class as a mechanism to filter log entries by area. Log classes in combination with log levels allow you to troubleshoot using log entries that only address, for example, configuration.

Log classes (as well as log levels) can be filtered at run time using `egosh debug`.

Valid logging classes are as follows:

Log class	Description
LC_TRACE	Logs significant program steps.
LC_COMM	Logs messages related to communications.
LC_XDR	Logs everything transferred by XDR
LC_FILE	Logs file transfer messages.

Log class	Description
LC_AFS	Logs AFS messages.
LC_AUTH	Logs messages related to users and authentication.
LC_HANG	Marks where a program might hang.
LC_SIGNAL	Logs messages pertaining to signals.
LC_DCE	Logs messages pertaining to DCE support.
LC_PIM	Logs PIM messages.
LC_MEMORY	Logs memory limit messages.
LC_SYS	Logs system call messages.
LC_EEVENTD	Logs eeventd messages.
LC_LOADINDX	Logs load index messages.
LC_RESOURCE	Logs information used by resource broker (resource gathering and reporting).
LC_M_LOG	Logs multievent logging messages.
LC_PERFM	Logs performance messages.

Log levels

Use `EGO_LOG_MASK` to specify the log level. For example, `EGO_LOG_MASK=LOG_CRIT`.

For most logs, there are nine log levels that allow administrators to control the level of event information that is logged. For logs associated with the reporting feature, there are seven log levels.

When you are troubleshooting, increase the log level to obtain as much detailed information as you can. When you are finished troubleshooting, decrease the log level to prevent the log files from becoming too large and to enhance daemon performance.

Valid logging levels are as follows (not including the reporting feature log levels):

Log level	Description
LOG_EMERG	Logs only those messages in which the system is unusable.
LOG_ALERT	Logs those messages for which action must be taken immediately.
LOG_CRIT	Logs those messages that are critical.
LOG_ERR	Logs those messages that indicate error conditions.
LOG_WARNING	Logs those messages that are warnings or more serious messages. This is the default level of debug information.
LOG_NOTICE	Logs those messages that indicate normal but significant conditions or warnings and more serious messages.
LOG_INFO	Logs all informational messages and more serious messages.
LOG_DEBUG	Logs all debug-level messages.

Log level	Description
LOG_TRACE	Logs all available messages. **Note:** LOG_TRACE is not supported by the LIM. If you set LOG_TRACE for the LIM, it is automatically changed to LOG_DEBUG.

Valid log levels for reporting feature are as follows:

Log level	Description
OFF	Logs no messages.
FATAL	Logs messages that were fatal to the reporting feature.
ERROR	Logs those messages that indicate error conditions.
WARN	Logs those messages that are warnings or more serious messages.
INFO	Logs all informational messages and more serious messages. (Default)
DEBUG	Logs all debug-level messages
ALL	Logs all messages.

EGO log file properties

Log properties in the Console include the following (as seen on the Logs (List) page):

Property	Definition
Category name	Identifies the log name.
Summary/Description	Provides a brief summary of the log purpose and the information that is written to the log file.
System Component	Identifies from what product system component this log originates from. For example, the lim logs originate in the EGO component.

View EGO log files

Event and audit log files can be viewed through the Platform Management Console.

1. From the Platform Management Console, navigate to System Logs.
2. Select Standard Logs.
3. From Logs (List), select a log from the list (for example, “lim.log”).

   Logs of the selected category are listed for every management host in the cluster.
4. Under Log retrieval parameters, specify the following:
   a) Optional. Filter the available hosts by typing full or partial host names in the Filter available hosts field.

      The list of available hosts is dynamically filtered as you type to show those hosts containing the same characters/name.

   b) Choose which hosts you want to retrieve and view this log from (for example, all hosts for which you want to view “lim.log” from).
      1. Click a host name from the Available hosts list.
      2. Click Add - > to move this host to the Retrieve logs from these hosts list.
      3. Do this for as many hosts as you want to retrieve logs from.

      Note:

      For performance reasons, there is a limit to the number of hosts you can add to the list. A message alerts you when this limit is reached.
5. Click Retrieve Log List.
6. Specify how much of the retrieved log you want to see. Options for retrieved data volume include the following:
   - Complete log
   - Last number of lines (a specified number of lines in the file); if you choose to retrieve a specified number of lines, enter the number in the field beside the drop-down list

     Note:

     You only see the last number of specified lines in the file, not the first.

   A list of logs from the requested hosts displays, along with details including the host on which the log file is located, the log file size, and the date of the last entry in the log file.
7. Click a specific log file to view or save.

   Choose to open and view the log file in a specified program, or save it to disk.

   Note:

   Log files cannot be edited, renamed, or deleted from within the Platform Management Console.

Change EGO log levels

Log on as root or egoadmin.

If you need to troubleshoot or debug your system, you need to change log levels from the default LOG_WARNING to a log level that gives more details (we suggest LOG_DEBUG). You can also enable or disable logging completely.

Separate log levels are set for each daemon. To change the log level for an entire cluster, you must set each daemon individually.

Logging level settings are not permanent and do not change `ego.conf`. When the cluster shuts down or restarts, logging level settings do not persist. Instead, log levels reset to LOG_WARNING.

Refer to the egosh command for details on all the command options.

Note:

If you are setting the log level for VEMKD and this daemon fails over, the permanent setting in ego.conf (Warning level) becomes the current setting. To change it, follow these steps once more.

1. To enable the more detailed log level LOG_DEBUG, run `egosh debug` *daemon* `on`.

   For example, `egosh debug vemkdon`.

   You can also specify a log class using the -c option if you want to receive entries solely concerning one log class (like authentication or memory). Refer to the `egosh` command for details on using this option.

   You have enabled log level LOG_DEBUG for vemkd. All entries are logged in `vemkd.log.`*`hostname`*.
2. To return the log level to the default LOG_WARNING, run `egosh debug` *daemon*`off`.

   For example, `egosh debug vemkdoff`.

   You have returned your log level to LOG_WARNING (the default) for vemkd. All entries are logged in `vemkd.log.`*`hostname`*.

You should debug your system as quickly as possible and set your log level to warning for daily activity.

Caution:

The LOG_DEBUG logging level is very verbose and fills up your log files quickly, which can adversely affect performance.

Change EGO log file locations

On Windows machines, you can save log files to a specified location. If you are running a clustered application manager (such as Platform LSF), you want to ensure the location is the same for both applications (both LSF and EGO).

1. Open `%EGO_CONFDIR%\ego.conf`.
2. To enable logging in Windows, you must define the parameter EGO_LOGDIR=*dir*, where *dir* specifies the EGO system log file directory.

   Error messages from all servers are logged into files in this directory. To effectively use debugging, set EGO_LOGDIR to a directory such as `C:\Temp`.
3. To enable logging of error messages into the directory files specified by EGO_LOGDIR, you must also define the related parameter EGO_LOGDIR_USE_WIN_REG=**n**.

   If you do not define EGO_LOGDIR_USE_WIN_REG, or if you define it with a value other than **n**, then EGO logs error messages to the default local directory specified in the Windows registry key:

   HKEY_LOCAL_MACHINE\SOFTWARE\Platform Computing Corporation\EGO\EGO_LOGDIR

   Note:

   For 64-bit Windows machines, EGO logs error messages to the default local directory specified in this Windows registry key: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Platform Computing Corporation\EGO\EGO_LOGDIR.

   If a server is unable to write in the EGO system log file directory, EGO attempts to write to these directories:

   - Linux/UNIX, in this order:
     1. $TMPDIR (if variable is defined)
     2. /tmp
   - Windows, in this order:
     1. OS default temp directory, by call Windows API GetTempPath()
     2. System directory (C:\)

Change Symphony log levels

You can change log levels dynamically using the `soamlog` command. This is useful to debug active processes.

Changes made with the `soamlog` command are temporary. They remain in effect until the host on which the process is running is restarted, the process is closed, or the application is closed.

Symphony uses the log4j logging framework. Log classes can be found in the log4j properties files located in the conf directory located under SOAM_HOME.

1. Use the `soamlog` command to change the log level of Symphony components.

   For example:

   - To change the workload trace level to `Warning`:

     ```
     soamlog workload sampleApp -l LOG_WARN
     ```
   - To change all the log levels for session manager components for application `sampleApp` to `Warning`:

     ```
     soamlog ssm_all sampleApp -l LOG_WARN
     ```
   - To change all the log levels of service instance managers working for application `sampleApp` to `Error`:

     ```
     soamlog sim_all sampleApp -l LOG_ERROR
     ```
   - To change session director log levels, you must be the cluster administrator. To change all the log levels of session director components to `Debug`:

     ```
     soamlog sd_all -l LOG_DEBUG
     ```

Change time zone settings on Linux/UNIX

On Linux/UNIX, by default, Symphony components use GMT time when messages are logged regardless of the local time setting on Linux/UNIX machines.

To change the time zone to your local time setting, change the `*component_name*.log4j.properties` files on each Linux/UNIX host in the `$SOAM_HOME/conf` directory.

For example, to change the log4j.properties file for session manager:

1. Open the `ssm.log4j.properties` file.
2. Uncomment one of the following lines to set your time zone:

```
#log4j.appender.SSM.layout.TimeZone=Canada/Eastern
#log4j.appender.SSM.layout.TimeZone=PRC
#log4j.appender.SSM.layout.TimeZone=Europe/Paris
#log4j.appender.SSM.layout.TimeZone=Europe/London
#log4j.appender.SSM.layout.TimeZone=EST
```

3. Uncomment one of the following lines to set the output format with your time zone.

```
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q Canada/
Eastern} %p [${log4cxx_pid}:%t] %c - %m%n
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q PRC} %p [$
{log4cxx_pid}:%t] %c - %m%n
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q Europe/Paris}
%p [${log4cxx_pid}:%t] %c - %m%n
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q Europe/
London} %p [${log4cxx_pid}:%t] %c - %m%n
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q EST} %p [$
{log4cxx_pid}:%t] %c - %m%n
```

4. Save the file.

   Your file should now look similar to the following. In this example, we have uncommented lines to set the time zone to EST:

```
...
#Pattern to output the caller's file name and line number.
log4j.appender.SSM=org.apache.log4j.RollingFileAppender
#log4j.appender.SSM.File=${SOAM_HOME}/logs/ssm.${log4cxx_hostname}.log
log4j.appender.SSM.File=${SOAM_HOME}/logs/ssm.${log4cxx_hostname}.$
{log4cxx_appname}.log
log4j.appender.SSM.MaxFileSize=100000KB

#Time Zone Setting
#choose your time zone
#for example, for Canada Eastern time, use Canada/Eastern or EST
#log4j.appender.SSM.layout.TimeZone=Canada/Eastern
#log4j.appender.SSM.layout.TimeZone=PRC
#log4j.appender.SSM.layout.TimeZone=Europe/Paris
#log4j.appender.SSM.layout.TimeZone=Europe/London
log4j.appender.SSM.layout.TimeZone=EST

# Keep one backup file
log4j.appender.SSM.MaxBackupIndex=10
log4j.appender.SSM.layout=org.apache.log4j.PatternLayout
#if you use the timezone setting in Windows, please use this pattern
log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q %Z} %p [$
{log4cxx_pid}:%t] %c - %m%n
#if you use the timezone setting in Linux, please use this pattern (replace %Z with
your time zone)
#for example, if your time zone is EST, replace %Z with EST, if other time zones,
please replace %Z with others
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q Canada/
Eastern} %p [${log4cxx_pid}:%t] %c - %m%n
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q PRC} %p [$
{log4cxx_pid}:%t] %c - %m%n
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q Europe/Paris}
%p [${log4cxx_pid}:%t] %c - %m%n
```

```
#log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q Europe/
London} %p [${log4cxx_pid}:%t] %c - %m%n
log4j.appender.SSM.layout.ConversionPattern=%d{%Y-%m-%d %H:%M:%S.%Q EST} %p [$
{log4cxx_pid}:%t] %c - %m%n
..
```

About log files and levels

Use the Symphony log files to troubleshoot workload related components such as session director, session manager, and service instance manager.

Log files

The Symphony log files provide information on the general well-being of workload-related daemons and services.

Default log file locations

Symphony component log files

- Windows—`%SOAM_HOME%\logs`
- Linux/UNIX—`$SOAM_HOME/logs`

Symphony API log files

The Symphony API log file is written to the directory where the client executable resides.

Log file names

Log files are named according to the component they are logging and the host name where the component runs. For example, a log file for the session director running on hostA is named `sd.hostA.log`.

The following table lists possible log files and on which hosts they can be found.

Log file	Description	Host on which you can find the log file
sd.*host_name*.log	Messages, events, and errors for session director.	The host on which session director is running.
ssm.*host_name.application_name*.log	Messages, events, and errors related to workload scheduling for the specified application.	The host on which session manager is running.
sim.*host_name.application_name*.log	Messages, events, and errors related to tasks that ran for the specified application.	Each compute host running tasks for the application.
api.*host_name*.log	Messages, events, and errors for the client application that submits work to the system.	The host on which the client application runs.
agent.*host_name*.log	Only in Symphony DE. Messages, events, and errors related to startup and shut down of Symphony DE processes.	Only in Symphony DE. Found on every host on which Symphony DE runs.

Log file	Description	Host on which you can find the log file
cli.log	Messages, events, and errors related to the command line.	When enabled, found on the host from which the command was issued, in the directory from which the command was issued. By default, no `cli.log` files exist.

Logging configuration files

Default properties file location

The default locations of the logging configuration (properties) files are:

- Windows—`%`*`SOAM_HOME`*`%\conf`
- Linux/UNIX—`$`*`SOAM_HOME`*`/conf`

Available properties files

The following properties files are available:

- `agent.log4j.properties`
- `api.log4j.properties`
- `cli.log4j.properties`
- `rs.log4j.properties`
- `sd.log4j.properties`
- `sim.log4j.properties`
- `ssm.log4j.properties`

Log file formats

Log file entries follow a format that depends on the log level in which the message was logged.

File format customization

The format of the log-file entries can be changed. For more details, see the log4cxx documentation: `http://logging.apache.org/log4cxx/manual/classlog4cxx_1_1PatternLayout.html`

Synopsis for INFO log level

time_stamp log_level [*process_ID*:*thread_ID*] *logger_name* - *info_message*

Synopsis for WARN, ERROR, and FATAL log level messages

time_stamp log_level [*process_ID*:*thread_ID*] *logger_name* - Code[*Internal_Code*]: *file_name*:*line_number message*

Log file attributes

The following information is included for all messages recorded at the INFO, WARN, ERROR, and FATAL log levels:

```
time_stamp log_level [process_ID:thread_ID] logger_name
```

The following information is included for some errors:

```
Code[Internal_Code]: file_name:line_number
```

The remainder is the main body of the message. It can include information such as error domain, consumer ID, command, workDir, and hostname, along with the message.

Attributes of INFO, WARN, DEBUG, ERROR, and FATAL log level messages

time_stamp

Displays the time when the exception was thrown. The format for the time stamp is:

```
Year-month-day hour:minute:second.millisecond
```

Note:

For Linux/UNIX users only—By default, the time displayed in the logger files is GMT. The format of the timestamp can be changed by editing the related time zone settings in each `log4j.properties` file in $SOAM_HOME/conf. The properties files contain instructions on how to implement this change.

log_level

Displays the log level of the logger that logged the message.

Level	Description
FATAL	Logs only those messages in which the system is unusable.
ERROR	Logs only those messages that indicate error conditions or more serious messages.
WARN	Logs only those messages that are warnings or more serious messages. This is the default level of debug information.
INFO	Logs all informational messages and more serious messages.
DEBUG	Logs all debug-level and INFO messages.
ALL	Logs all available messages.

process_ID

Displays the ID of the Symphony component. The process ID is used to differentiate between daemons when more than one daemon of the same type runs on the host, such as when multiple session managers run on the same host.

Note:

The `soamview app` command displays the process ID of the session manager and `soamview task` displays the process ID of the service instance. The identity of the process that generated the message can be determined by comparing the process ID in the message with the process IDs displayed by `soamview app` and `soamview task`.

thread_ID

Displays the thread of the program that triggered the message.

logger_name

Displays the name of the logger component used to set the log level of the component that generated the message. The log4j logger components are listed in the properties files. These loggers are used to set the logging levels of specific components such as session director, session manager, service instance manager, repository service, and the CLI.

error_message

Displays the error message generated by the Symphony API.

error_code

Displays the error code that uniquely identifies the error. Error codes and their corresponding messages are listed in the *Error Message Reference*.

file_name

Displays the name of the source code file that triggered the message.

line_number

Displays the number of the line in the file that triggered the message.

domain

Displays the domain in which the message was triggered. Domains are virtual groups that categorize messages to precisely identify the component the message applies to.

Possible domains are:

- Application—Application configuration and deployment
- SOAM—Any Symphony component such as session manager and session director
- VEM—Resource management performed by EGO (not available in Symphony DE)
- OS—Operating system resource management of resources such as memory and disk capacity

CHAPTER

25

Events

Symphony events

Platform Symphony events can be monitored for and used to trigger actions automatically.

Platform Symphony events are categorized as follows:

- EGO system events, which identify host- and service-related occurrences within the cluster
- SOAM system events, which identify session-manager related occurrences within the cluster
- Application events, which identify occurrences that affect workload
- Platform Management Console events, which identify occurrences that affect the web server or the Platform Management Console itself.

By default, events are not enabled. If you want to be notified about application events, you need to enable the event framework. Even if the event framework is not enabled, you can monitor events in the Symphony log files in the logs directory.

EGO system events

Event name	Default level	Triggered when ...
SYS_CLS_UNLICENSED • Component name: ego_lim • Returned integer: 0	Error	The cluster is not licensed
SYS_HOST_CLOSED • Component name: ego_vemkd • Returned integer: 6	Warning	A host is closed
SYS_HOST_UNAVAIL • Component name: ego_vemkd • Returned integer: 4	Warning	A host becomes unavailable
SYS_PEM_DOWN • Component name: ego_lim • Returned integer: 2	Error	Local pem goes down
SYS_PEM_UP • Component name: ego_pem • Returned integer: 13	Info	Local pem is started
SYS_SVC_DOWN • Component name: ego_sc • Returned integer: 7	Error	A system service goes down

Event name	Default level	Triggered when …
SYS_SVC_UP • Component name: ego_sc • Returned integer: 8	Info	A system service is started
SYS_SVC_INST_DOWN • Component name: ego_sc • Returned integer: 9	Error	An instance of a system service goes down
SYS_SVC_INST_UP • Component name: ego_sc • Returned integer: 10	Info	An instance of a service is started
SYS_VEMKD_DOWN • Component name: ego_lim • Returned integer: 1	Error	vemkd goes down
SYS_VEMKD_UP • Component name: ego_vemkd • Returned integer: 11	Info	vemkd is started

SOAM system events

Event Name	Default level	Triggered when …
SYS_BM_BOUNDARY_BREACHED	Warning	The session manager memory usage exceeds threshold (%).
SYS_DS_READFAIL_SESSION SYS_DS_READFAIL_TASKINPUT SYS_DS_READFAIL_TASKOUTPUT SYS_DS_READFAIL_CDU	Error	The session manager failed to read from data storage.
SYS_DS_WRITEFAIL_SESSION SYS_DS_WRITEFAIL_TASKINPUT SYS_DS_WRITEFAIL_TASKOUTPUT SYS_DS_WRITEFAIL_SESSION_OBJECT SYS_DS_WRITEFAIL_TASK_OBJECT SYS_DS_WRITEFAIL_CDU SYS_DS_WRITEFAIL_CDU_OBJECT	Warning	The session manager failed to write to data storage.
SYS_FAILOVER_RETRIED	Info	Trying to restart the session manager or service instance manager.

Event Name	Default level	Triggered when …
SYS_SSM_DOWN	Info	The session manager goes down abnormally.
SYS_SSM_UP	Info	The session manager comes up.

Application events

Event Name	Default level	Triggered when …
SOA_SERVICE_BLOCKED	Error	A service instance is blocked from a host.
SOA_SERVICE_CUSTOM_ACTION	Error	A service instance returns a particular code.
SOA_SERVICE_DEPLOYMENT_FAILED	Error	A service failed to deploy.
SOA_SERVICE_EXITED	Error	A service instance exited.
SOA_SERVICE_FAILURE	Error	A service instance threw a failure exception.
SOA_SERVICE_FATAL_ERROR	Error	A service instance threw a fatal exception.
SOA_SERVICE_INIT_FAILED	Error	A service instance creation failed.
SOA_SERVICE_RUNAWAY	Error	A service instance takes longer than expected to complete.
SOA_SESSION_ABORTED	Error	A session is aborted.
SOA_SESSION_LOST	Error	A lost connection from the session is detected.
SOA_SESSION_PRI_CHANGED	Info	The priority of a session is changed and the session is resumed.
SOA_SESSION_RESUMED	Info	A session is resumed.
SOA_SESSION_SUSPENDED	Warning	A session is suspended.
SOA_TASK_EXIT	Error	A task exited, such as when a service instance crashes.
SOA_TASK_FAILURE	Error	A service instance threw a failure exception during the invoke call.
SOA_TASK_FATAL_ERROR	Error	A service instance threw a fatal exception during the invoke call.
SOA_TASK_RUNAWAY	Error	A task runs longer than expected and a timeout is invoked.

Platform Management Console events

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

Enable events

Log on as root or egoadmin

If you want to be notified about cluster events, enable this feature. By default, events are not enabled.

1. Edit `ego.conf`.

   You need to specify two parameters under the EGO event configuration section: EGO_EVENT_PLUGIN and EGO_EVENT_MASK. These parameters are already included in `ego.conf` with default values, but may be commented out.
2. Set up an SNMP v1 (Simple Network Management Protocol version 1) trap for EGO events.
   a) Specify the name and configuration file location with your SNMP information. The plug in name should not include a suffix (.dll or .so): EGO_EVENT_PLUGIN=plugin_name[plugin_conf]…

      Example:

      `EGO_EVENT_PLUGIN=eventplugin_snmp[SINK=`*host*,MIBDIRS=*EGO_TOP*/mibs]

      (where *host* represents the name of the host where the SNMP trap daemon is running).

      SNMP traps enable an agent to notify the management station of significant events by way of an unsolicited SNMP message.

      Note the following:

      - The `MIBDIRS` directory may also equal *`$EGO_CONFDIR`*`/kernel/conf/mibs`.
      - In a Windows environment, use quotation marks around the event plug-in definition.

        For example, `EGO_EVENT_PLUGIN="eventplugin_snmp [SINK=`*host*,MIBDIRS=*EGO_TOP*\mibs]"
   b) You can modify the default port (port 162) by specifying `TRAPPORT=`*`port_number`* in the string.

      `EGO_EVENT_PLUG_IN="eventplugin_snmp[…,TRAPPORT=`*`port_number`*`]"`

      Note that in a Linux/UNIX environment, do not use quotation marks around the event plug-in definition.
3. Set EGO_EVENT_MASK to the log level you want.

   EGO_EVENT_MASK must be set to one of the following values:

   - LOG_ERR: Provides information about error events only
   - LOG_WARNING: Provides information about warning and error events
   - LOG_INFO: (Default) Provides information about all events

   For example, **EGO_EVENT_MASK=LOG_INFO**.
4. Save `ego.conf` and restart your cluster.
5. Using your SNMP Manager, select what actions it takes when it receives a specific trap. See your SNMP Manager help for more information.

Responding to event message sys_cls_unlicensed

This is a system event triggered when the cluster is not licensed. By default, this event is set to an Error level.

1. Ensure that the EGO_LICENSE_FILE variable in `ego.conf` is porting to the correct file location on the master host.
2. Review the license file content and ensure it is valid. Confirm that the license has not expired.

CHAPTER

26

Audit Logs

EGO audit logs

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

Specify the location of audit log files with the parameter `EGO_AUDIT_LOGDIR` in `ego.conf`. To enable audit logs, set `EGO_AUDIT_LOG=Y` in `ego.conf`.

Component/ object	Logged event/action	Audit log file name

EGO service	- Start
	- Stop
		Windows:
		`%EGO_CONFDIR%\..\..\audits \egoservice.audit.log`
		Linux/UNIX:
		`$EGO_CONFDIR/../../audits/ egoservice.audit.log`

Host	- Open
	- Close
		Windows:
		`%EGO_CONFDIR%\..\..\audits\ego.audit.log`
		Linux/UNIX:
		`$EGO_CONFDIR/../../audits/ego.audit.log`

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

Consumer	- Add
	- Modify
	- Delete
	- Change resource plan
		Windows:
		`%EGO_CONFDIR%\..\..\audits\ego.audit.log`
		Linux/UNIX:
		`$EGO_CONFDIR/../../audits/ego.audit.log`

Audit log file format

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

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

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

Note:

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

Note:

With the exception of egosh user logon and egosh user logoff, three events are logged for commands: logon, the command, logoff. This is because authentication is required for command-line interfaces. For the command egosh user logon and egosh user logoff, only two events are logged: logon and logoff.

Application-related audit logs

Application-related audit logs allow you to track user operations on Symphony services managed by EGO, service packages, sessions, and applications.

Note:

Audit logs do not exist in Symphony DE.

Audit log file	Object	Audited operation
repositoryservice.audit.log	Service packages	• Add • Remove
application.audit.log	Sessions	• Kill • Suspend • Resume
application.audit.log	Applications	• Enable • Disable • Register • Unregister
egoservice.audit.log	Services managed by EGO: • Session Director • Repository Service **Note:** EGO Service Controller event logging must be turned on for service events to be logged	• Start • Stop

Log location

Application-related audit logs are located in the same directory as EGO audit logs and the location is defined with the parameter EGO_AUDIT_LOGDIR in the configuration file `ego.conf`.

Audit log file format

The log file format for application-related audit log files is the same as that of the EGO audit log files with additional possible objects and actions. The table below lists the additional objects and actions in context of the audit log file format.

Note:

The service object already exists in EGO. What is additional for applications is the logging of the SD and RS service actions.

DATE/TIME	TYPE	USER	OBJECT	ID	ACTION	DETAIL
time_stamp	CONTROL	user_name	PACKAGE	application_name	added	-
time_stamp	CONTROL	user_name	PACKAGE	application_name	removed	-
time_stamp	CONTROL	user_name	PACKAGE	application_name	add fail	*error_msg*
time_stamp	CONTROL	user_name	PACKAGE	application_name	remove fail	*error_msg*
time_stamp	CONTROL	user_name	APPLICATION	application_name	enabled	-
time_stamp	CONTROL	user_name	APPLICATION	application_name	disabled	-
time_stamp	CONTROL	user_name	APPLICATION	application_name	registered	-
time_stamp	CONTROL	user_name	APPLICATION	application_name	unregistered	-
time_stamp	CONTROL	user_name	APPLICATION	application_name	enable fail	*error_msg*
time_stamp	CONTROL	user_name	APPLICATION	application_name	disable fail	*error_msg*
time_stamp	CONTROL	user_name	APPLICATION	application_name	register fail	*error_msg*
time_stamp	CONTROL	user_name	APPLICATION	application_name	unregister fail	*error_msg*
time_stamp	CONTROL	user_name	SESSION	application_name:session_ID	killed	-
time_stamp	CONTROL	user_name	SESSION	application_name:session_ID	suspended	-
time_stamp	CONTROL	user_name	SESSION	application_name:session_ID	resumed	-
time_stamp	CONTROL	user_name	SESSION	application_name:session_ID	kill fail	*error_msg*
time_stamp	CONTROL	user_name	SESSION	application_name:session_ID	suspend fail	*error_msg*
time_stamp	CONTROL	user_name	SESSION	application_name:session_ID	resume fail	*error_msg*

`DATE/TIME`	`TYPE`	`USER`	`OBJECT`	`ID`	`ACTION`	`DETAIL`
time_stamp	CONTROL	user_name	SERVICE	SD	started	-
time_stamp	CONTROL	user_name	SERVICE	SD	stopped	-
time_stamp	CONTROL	user_name	SERVICE	SD	start_failed	*error_msg*
time_stamp	CONTROL	user_name	SERVICE	SD	stop_failed	*error_msg*
time_stamp	CONTROL	user_name	SERVICE	RS	started	-
time_stamp	CONTROL	user_name	SERVICE	RS	stopped	-
time_stamp	CONTROL	user_name	SERVICE	RS	start_failed	*error_msg*
time_stamp	CONTROL	user_name	SERVICE	RS	stop_failed	*error_msg*

Enable EGO event logging for auditing purposes

You must be a cluster administrator to perform this task. You have previously run `egoconfig mghost` *shared_dir* during the installation for multi-host clusters or Linux/UNIX installations.

EGO monitors and logs security-sensitive events related to EGO services and to host, user, and consumer containers. By default, auditing of these events is disabled. To collect information to better monitor system security, enable logging by configuring the `ego.conf`, `egosc.conf.xml`, and `rs.xml` files.

Important:

Linux/UNIX installations require a shared directory.

Note the following:

- Audit logs can be enabled independently of each other.
- For multi-host clusters, configure files from within the shared directory, not the local directory (local directory configurations are ignored). For single-host Windows clusters, you can configure local files.
- Only master hosts perform audit logging; compute hosts do not normally have access to the shared locations where configuration files are stored. You never need to enable audit logging (configure files) on compute hosts.

1. To enable logging for auditing of core EGO functions (for example, security):
   a) Open `ego.conf`.
      - On Windows: `%EGO_CONFDIR%\ego.conf`
      - On Linux/UNIX: `$EGO_CONFDIR/ego.conf`

   b) Turn on EGO audit logging by adding the following parameter:

      EGO_AUDIT_LOG=**Y**

   c) At this time, you may also want to define an audit log directory by configuring the `EGO_AUDIT_LOGDIR` parameter. This is the default directory location and name:
      - On Windows: `EGO_AUDIT_LOGDIR=%EGO_CONFDIR%\audits`
      - On Linux/UNIX: `EGO_AUDIT_LOGDIR=$EGO_CONFDIR/audits`

      Note:

      You can change the name, but the location must be a shared directory; ensure there are no spaces in the directory name.

      Once defined, the vemkd and egosc daemons automatically create the directory.

   d) Save and close the file.

      Note that there is no automatic file roll-over or audit log cleanup. Ensure that you manually manage the file size.
2. To enable audit logging for the service controller:
   a) Open `egosc.conf.xml`
      - On Windows: `EGO_TOP\\eservice\esc\conf\egosc.conf.xml`
      - On Linux/UNIX: `EGO_TOP//eservice/esc/conf/egosc.conf.xml`

   b) Turn on the EGO service controller log (`egoservice.audit.log`) by adding the following element:

      <ESC_AUDIT_LOG>**ON**</ESC_AUDIT_LOG>

   c) Save and close the file.

3. To enable audit logging for the repository service:
   a) Open `rs.xml`.
      - On Windows: *`EGO_TOP`*`\eservice\esc\conf\services\rs.xml`
      - On Linux/UNIX: *`EGO_TOP`*`/eservice/esc/conf/services/rs.xml`

   b) Turn on the repository service audit log by adding the following element to the `ego:ActivitySpecification` sections for all OS types:

      ```
      <ego:EnvironmentVariable name="RS_AUDIT_LOG">ON</ego:EnvironmentVariable
      >
      ```

      Note:

      The default setting is OFF. The setting is case sensitive.

      The RS logs information into the configured audit log directory, as specified by the parameter EGO_AUDIT_LOGDIR defined in `ego.conf`. If this parameter is not found or defined, the RS logs to this directory:
      - On Windows: *`EGO_TOP`*`\audits`
      - On Linux/UNIX: *`EGO_TOP`*`/audits`

   c) Save and close the file.

   d) Stop the RS service.

      egosh service stop RS

4. Restart EGO on the master host.

   egosh ego restart

   EGO restarts any currently stopped services. Changes made to stopped services now take effect.

Enable application-event logging for auditing purposes

You must be a cluster administrator to perform this task.

Application audit logs allow you to track user operations on Symphony services managed by EGO, service packages, sessions, and applications.

By default, auditing is not enabled.

Note:

Auditing is not supported in Symphony DE.

---

1. Enable auditing of operations performed on service packages.
    a) Log on to the master host in the cluster.
    b) Open the `rs.xml` configuration file, located in the `eservice` directory under the directory in which Symphony was installed.

        For example, on Windows

        %EGO_CONFDIR%\..\..\eservice\esc\conf\services\rs.xml

        For example, on Linux/UNIX

        $EGO_CONFDIR/../../eservice/esc/conf/services/rs.xml
    c) Add the RS_AUDIT_LOG element in `<ego:ActivitySpecification>`.

```
<sc:ServiceDefinition xmlns:sc="http://www.platform.com/ego/2005/05/schema/sc" xmlns:ego="http://
www.pla/www.platform.com/ego/2005/05/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://
www.platform.com/ego/2005/05/schema/sc ../sc.xsd http://www.platform.com/ego/2005/05/schema ../
ego.xsd" ServiceName="RS">
 ...
<sc:ActivityDescription>
<ego:Attribute name="hostType" type="xsd:string">NTX86</ego:Attribute>
 <ego:ActivitySpecification>
 ...
  <ego:EnvironmentVariable name="RS_AUDIT_LOG">ON</ego:EnvironmentVariable>
 </ego:ActivitySpecification>
</sc:ActivityDescription>
...
```

    d) Save the file.
2. Enable auditing of operations performed on sessions and applications.
    a) Log on to the master host in the cluster.
    b) Open the `sd.xml` configuration file, located in the `eservice` directory under the directory in which Symphony was installed.

        For example, on Windows

        %EGO_CONFDIR%\..\..\eservice\esc\conf\services\sd.xml

        For example, on Linux/UNIX

        $EGO_CONFDIR/../../eservice/esc/conf/services/sd.xml
    c) Add the SD_AUDIT_LOG element in `<ego:ActivitySpecification>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<sc:ServiceDefinition xmlns:sc="http://www.platform.com/ego/2005/05/schema/sc"
xmlns:ego="http://www.pla/www.platform.com/ego/2005/05/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://www.platform.com/ego/
```

```
2005/05/schema/sc ../sc.xsd http://www.platform.com/ego/2005/05/schema ../
ego.xsd" ServiceName="SD">
...
<sc:ActivityDescription>
...
 <ego:ActivitySpecification>
...
   <ego:EnvironmentVariable name="SD_AUDIT_LOG">ON</ego:EnvironmentVariable>
 </ego:ActivitySpecification>
</sc:ActivityDescription>
...
```

d) Save the file.

3. Log on to the master host and stop and restart the Session Director and Repository Service.

   a) Restart EGO on the master host.

      egosh ego restart

   b) Stop the services:

      egosh service stop RS

      egosh service stop SD

   c) Start the services:

      egosh service start RS

      egosh service start SD

CHAPTER

27

Traces

Traces

Traces enable an administrator to turn logging on for information specific to an object in EGO. Currently, you can only set traces for activities. An activity is a hosting environment for a service; it provides the context for a service.

Setting a trace results in log entries for that object at log level LOG_TRACE.

Set a trace

Log on as root or egoadmin.

If you need to troubleshoot a specific object in EGO, you can set a trace for it. Setting a trace logs entries for that object with log level LOG_TRACE.

1. Run `egosh debug` *`daemon`*`on -t -c` *`LOG_CLASS`* `-o "`*`key=value`*`"host.` For example:

   ```
   egosh debug pemon -t -c LC_CLIENT -o "activity=114" hostA
   ```

   Specifying the logging class is optional.
2. To turn off the trace, run `egosh debug` *daemon*off *host* -t -c *LOG_CLASS* -o "*key=value*". For example:

   ```
   egosh debug pemoff -t -c LC_CLIENT -o "activity=114" hostA
   ```

   Once you turn off the trace, the log level resets to its default.

CHAPTER

28

Basic troubleshooting

Log on to the Platform Management Console

You have set your environment on this host. You are logged into the operating system as `egoadmin`.

The Platform Management Console allows you to monitor, administer, and configure your cluster.

1. If you do not already know the web server URL, run **egosh client view GUIURL_1**.
   a) Beside DESCRIPTION, find the web server URL, and copy it.

      For example: `http://Host_W:8080/platform`.
2. Launch any web browser and paste the address of the web server URL.

   The format of the URL is `http://`*`host_name:port_number`*`/platform`
3. Log on to the Platform Management Console for the first time by specifying
   - User Name: Admin
   - Password: Admin

   For security in a production environment, we strongly recommend that you change the password of the Admin account.

Using Client Metadata for Troubleshooting

Client metadata relating to sessions is stored in session attributes and session history for troubleshooting purposes. Using the PMC or `soamview` CLI, or querying the reporting database, you can see the following information about the client associated with a session: version, OS user, IP address, or host.

- Client version is the version of Symphony library in use by the client at run time (not the version of Symphony used to compile the client).
- Client OS user name is the name of the OS user running the client.
- Client IP address is the IP address of the host running the client. In a multi-homed host, it is the IP address used to connect to the session.
- Client host name is the local host name of the host running the client. In a multi-homed host, it is the name of the host that connects to the session.

The information you see is always from the most recent client to be associated with the session, so if the client has disconnected and recovered, it may not be the same host or OS user that submitted the workload originally.

If the length of any attribute is longer than the maximum limit, the display is truncated.

Fix UNKNOWN or DEFAULT Matched Models and Matched Types

Fixing UNKNOWN Matched Type or Matched Model

A model or type UNKNOWN indicates the host or lim on the host is down. You need to take immediate action.

1. Start the host.
2. With root (Unix) or administrator (Windows) permission, run `egosh ego start` *host_name* to start up the load information manager (lim) on the host.

   You can specify more than one host name to start up the lim on multiple hosts. If you do not specify a host name, the `lim` is started up on the host from which the command is submitted.

   You must be a cluster administrator to run this command.

   On UNIX, to start up the lim remotely, you must be root or listed in `ego.sudoers` and be able to run the `rsh` command across all hosts without entering a password.
3. Wait a few seconds, then run `egosh resource view [resource_name …]`.

   You should now be able to see either a matched model or type for the host or the result DEFAULT. If you see DEFAULT, it means that automatic detection of host type or model has failed, and the host type configured in `ego.shared` cannot be found. EGO still works on the host, but there are disadvantages:

   - A DEFAULT Matched Type may cause binary incompatibility because a job from a DEFAULT host type can be migrated to another.
   - A DEFAULT Matched Model may be inefficient because of incorrect CPU factors.

Fixing DEFAULT Matched Type or Matched Model

If automatic detection of host type or model fails, and the host type configured in `ego.shared` cannot be found, then Matched Type gets set to DEFAULT. A Matched Type reported as DEFAULT may contribute to binary incompatibilities; a Matched Model reported as DEFAULT may be inefficient due to an incorrect CPU factor. You can run `lim -t` to detect the real type or model for a host, and then make changes to `ego.shared`.

1. Run `lim -t` on the host whose type is DEFAULT.
2. Edit `ego.shared`.
   a) In the `HostType` section, enter a new host type. Use the host type name detected with `lim -t`.
   b) In the `HostModel` section, add the new model with architecture and CPU factor. Add the host model to the end of the host model list. The limit for host model entries is 127. Lines commented out with # are not counted as part of the 127 line limit.

      Use the architecture detected with `lim -t`.
3. Save changes to `ego.shared`.
4. Run `egosh ego restart` on master host.
5. Wait a few seconds, then run `egosh resource view [resource_name …]` to check the type or model for a host.

Detect when failover occurs

You may need to know when a failover occurs so you can troubleshoot your system or your hosts.

1. Edit `ego.conf`.

   You need to specify two parameters under the EGO event configuration section: EGO_EVENT_PLUGIN and EGO_EVENT_MASK. These parameters are already included in `ego.conf` with default values, but may be commented out.
2. Set up an SNMP v1 (Simple Network Management Protocol version 1) trap for EGO events.
   a) Specify the name and configuration file location with your SNMP information. The plug in name should not include a suffix (.dll or .so): EGO_EVENT_PLUGIN=plugin_name[plugin_conf]…

      Example:

      `EGO_EVENT_PLUGIN=eventplugin_snmp[SINK=`*host*,MIBDIRS=*EGO_TOP*/mibs]

   (where *host* represents the name of the host where the SNMP trap daemon is running).

   SNMP traps enable an agent to notify the management station of significant events by way of an unsolicited SNMP message.

   Note the following:

   - The `MIBDIRS` directory may also equal *`$EGO_CONFDIR`*`/kernel/conf/mibs`.
   - In a Windows environment, use quotation marks around the event plug-in definition.

     For example, `EGO_EVENT_PLUGIN="eventplugin_snmp[SINK=`*host*,MIBDIRS=*EGO_TOP*\mibs]"
3. Set EGO_EVENT_MASK=LOG_INFO.

   SYS_VEMKD_UP (an INFO level log entry) occurs when you reconfigure your cluster or when your master host has successfully failed over to a new host.
4. Save `ego.conf` and restart your cluster.
5. Using your SNMP Manager, specify what action you want to take when the trap returns a SYS_VEMKD_UP event.

   You can select what actions the SNMP Manager takes when it receives a specific trap. See your SNMP Manager help for more information.

You are notified of a SYS_VEMKD_UP event that occurs only during successful failover and a cluster reconfiguration.

Note:

To verify that the master host has failed over, run `egosh resource list -m`. Your master candidates display. Check the Current Master entry.

Safely making configuration file changes

It is important and highly recommended to configure EGO through the Platform Management Console, not directly within the various configuration files.

Be aware of cause-effect relationships that exist in EGO between files. For example, manual changes made to the `ego.conf`, `ResourceGroup.xml` or `ConsumerTrees.xml` files may potentially affect other EGO configuration settings.

Furthermore, to achieve a specific goal, a whole set of related parameters must often be changed or tuned collectively, some in obscure directory locations. If not done correctly, the output of manually updated configuration files may not produce the expected behavior.

Finally, with certain configuration settings, important validations triggered through the Platform Management Console must be conducted by EGO. By using the Platform Management Console, legitimate and allowable parameter settings are ensured.

Examples of potential problems caused by manually configuring files:

- If you manually configure a consumer policy within “DistributionTree” sections of the `ConsumerTrees.xml` file without adding a corresponding instance in the “ConsumerHierarchy” section, then EGO does not recognize the newly added consumer.
- If you manually configure borrow and lend plans without giving a full consumer path (including the consumer tree name), then EGO may ignore them.
- If you set an invalid time window (not covering a 0 to 24 hour time period), then resource plans do not behave as expected.
- If you allot an unbalanced ownership, (such as reconfiguring a leaf consumer’s ownership value without changing the value at the branch level), then resource plans are not be effective and workload units do not run as expected.
- If you delete a consumer manually without checking whether it owns allocations or is currently running activities, you can seriously affect your resource plan and running workload units, among other things.

Why are tasks running on my master host?

When a cluster is not configured for failover, by default tasks can execute on any resource group (assuming that the default configuration for `resReq` and `ResourceGroupName` is used in `Consumer` section of the application profile).

When a cluster is configured for failover, by default work only runs on hosts that are not configured to be management hosts (hosts that are not marked with the `mg` resource attribute). Work does not run on any CPU slot in the `ManagementHosts` resource group.

For performance or other reasons you may not want workload to run on the master host. It is possible to change your configuration to dedicate the master host as a management host. In this way, you ensure no application workload runs on your master host.

Prevent tasks from running on your master host by configuring failover

1. You configure failover with the command `egoconfig mghost` *shared_top*. Running this command on the master host sets the shared directory for the cluster, copies configuration files to the shared directory, adds the host to the `ManagementHosts` resource group, and configures the `mg` resource attribute for the host.

Prevent tasks from running on your master host by manually changing the configuration

Log on to the host as the cluster administrator. For example, `egoadmin`.

If you do not want to configure failover for any reason, you can follow these steps to manually prevent tasks from running on your master host.

1. Open the cluster file on the master host for editing.

   Linux/UNIX—`$EGO_CONFDIR/ego.cluster.`*`cluster_name`*

   Windows—`%EGO_CONFDIR%\ego.cluster.`*`cluster_name`*
2. Find the `Host` section.

   ```
   Begin Host
   ...
   End Host
   ```
3. Locate your master host name and in the `RESOURCES` column add the `mg` resource attribute for your master host.

   This designates your master host as a management host only. Application workload will not run on this host.

   For example:

   ```
   Begin Host
   HOSTNAME  model    type        r1m  mem  swp  RESOURCES    #Keywords
   ...
   host1    !        NTX86       -    -    -    (nt mg)
   ...
   End Host
   ```
4. Save the file.
5. Restart EGO on the master host.

   ```
   egosh ego restart host1
   ```

Index