
Platform Symphony Foundations

Platform Symphony
Version 5.1
April 2011



Copyright

© 1994-2011 Platform Computing Corporation

All rights reserved.

Although the information in this document has been carefully reviewed, Platform Computing Corporation ("Platform") does not warrant it to be free of errors or omissions. Platform reserves the right to make corrections, updates, revisions or changes to the information in this document.

UNLESS OTHERWISE EXPRESSLY STATED BY PLATFORM, THE PROGRAM DESCRIBED IN THIS DOCUMENT IS PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL PLATFORM COMPUTING BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, DATA, OR SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

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

®LSF is a registered trademark of Platform Computing Corporation in the United States and in other jurisdictions.

™ACCELERATING INTELLIGENCE, PLATFORM COMPUTING, PLATFORM SYMPHONY, PLATFORM JOB SCHEDULER, PLATFORM ISF, PLATFORM ENTERPRISE GRID ORCHESTRATOR, PLATFORM EGO, and the PLATFORM and PLATFORM LSF logos are trademarks of Platform Computing Corporation in the United States and in other jurisdictions.

®UNIX is a registered trademark of The Open Group in the United States and in other jurisdictions.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

®Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Intel®, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other products or services mentioned in this document are identified by the trademarks or service marks of their respective owners.

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

1	Platform Symphony: An Overview	5
	Introduction to Platform Symphony	6
	Symphony cluster components	8
	Introducing the consumer	10
	Platform Symphony Developer Edition	11
	Platform Management Console	12
	Knowledge Center	14
	Reporting	15
	Security	16
2	Inside Workload Management	19
	SOA Middleware components	20
	Service-oriented application objects	22
	Application profile	23
	Consumers, applications, services, and service binaries	25
	Service package deployment	26
	Running executables in Symphony	27
3	Inside Resource Management	29
	EGO component overview	30
	Resources	32
	Consumers	34
	Resource sharing models	36
	Sharing of Symphony resources	38
4	Inside the Symphony Cluster	41
	Symphony processes	42
	Symphony cluster startup process	45
	Symphony fault tolerance	46
	Inside PERF	49

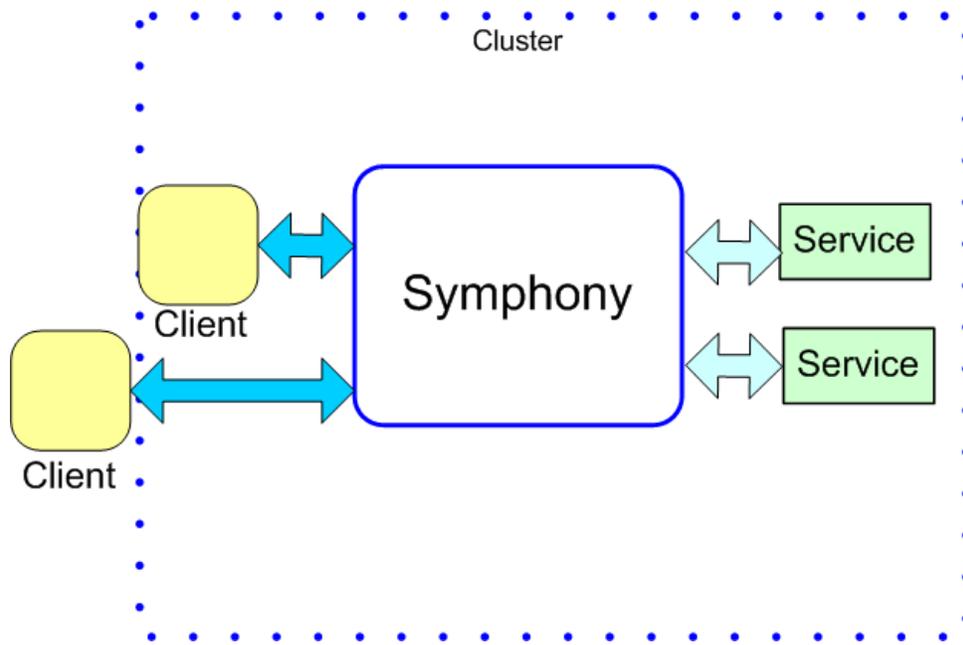
1

Platform Symphony: An Overview

Introduction to Platform Symphony

The Platform Symphony ("Symphony") software is leading enterprise-class software that distributes and virtualizes compute-intensive application services and processes across existing heterogeneous IT resources creating a shared, scalable, and fault-tolerant infrastructure, delivering faster, more reliable application performance while reducing cost.

Symphony provides an application framework that allows you to run distributed or parallel applications in a scaled-out grid environment.



Cluster

A cluster is a logical grouping of hosts that provides a distributed environment in which to run applications.

Symphony

Symphony manages the resources and the workload in the cluster. Using Symphony, resources are virtualized: Symphony dynamically and flexibly assigns resources, provisioning them and making them available for applications to use.

Symphony can assign resources to an application on demand when the work is submitted, or assignment can be predetermined and preconfigured.

Application

A Symphony service-oriented application uses a client/service architecture. It consists of two programs: the client, which provides the client logic to submit work, retrieve and process results, and the service, which comprises the business logic (the computation). The service-oriented application uses parallel processing to accelerate computations.

Symphony receives requests to run applications from a client. Symphony manages the scheduling and running of the work—the client need not be concerned with where the application runs.

Client

The client sends compute requests and collects results using the Symphony client APIs. The client may run on a machine that is part of the cluster, or it may run on a machine that is outside of the cluster.

The client can use a service without knowledge of what programming language was used to create the service.

The client submits an input data request to Symphony. Symphony initiates the service that processes the client requests, receives results from the service, and passes the results back to the client.

Service

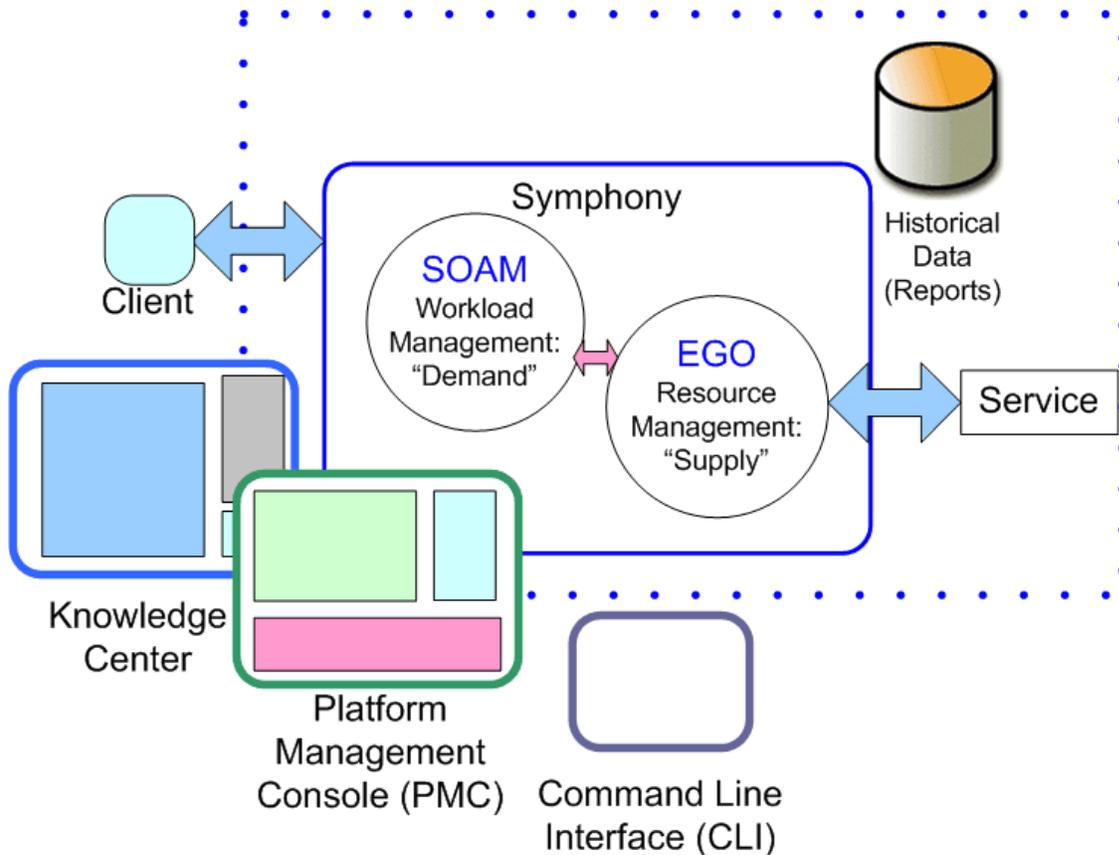
The service is a self-contained business function that accepts requests from a client, performs a computation, and returns responses to the client.

The service uses computing resources, and must be deployed to the cluster. Multiple instances of a service can run concurrently in the cluster.

The service is initiated and run by Symphony, upon receipt of a client request. The service runs on a machine that is part of the Symphony cluster. The service runs on the cluster resources dynamically provisioned by Symphony. Symphony monitors the running of the service, and passes the results back to the client.

Symphony cluster components

A Symphony cluster manages both workload and resources. Symphony maintains historical data, includes a web interface for administration and configuration, and also has a command-line interface for administration.



Workload management versus resource management

A workload manager interfaces directly with the application, receiving work, processing it, and returning the results. A workload manager provides a set of APIs, or may interface with additional run-time components to enable the application components to communicate and perform work. The workload manager is aware of the nature of the applications it supports using terminology and models consistent with a given class of workload. In a service-oriented application environment, workload is expressed in terms of messages, sessions, and services.

A resource manager provides the underlying system infrastructure to enable multiple applications to operate within a shared resource infrastructure. A resource manager manages the computing resources for all types of workload.

EGO--resource manager

Enterprise Grid Orchestrator ("EGO") manages the supply and distribution of resources, making them available to applications. EGO provides resource provisioning, remote execution, high availability, and business continuity.

EGO provides cluster management tools and the ability to manage supply versus demand to meet service-level agreements.

SOAM--workload manager

SOAM (SOA middleware) manages service-oriented application workload within the cluster, creating a demand for cluster resources.

When a client submits an application request, the request is received by SOAM. SOAM manages the scheduling of the workload to its assigned resources, requesting additional resources as required to meet service-level agreements. SOAM transfers input from the client to the service, then returns results to the client. SOAM releases excess resources to the resource manager.

Platform Management Console

The Platform Management Console (PMC) is your window to Symphony, providing resource monitoring capability, application service-level monitoring and control, and configuration tools.

Historical data for reporting

Symphony stores a wide variety of historical data for reporting and diagnostic purposes. Multiple reports capture and summarize the data.

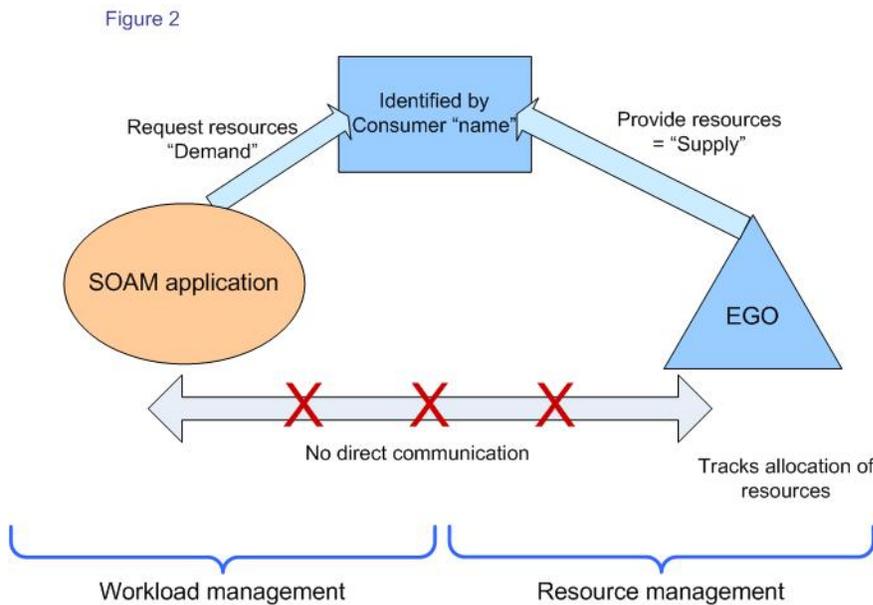
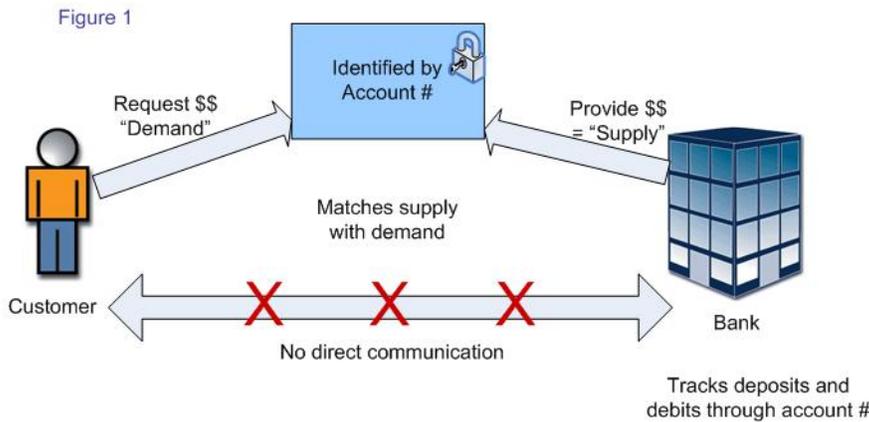
Knowledge Center

The Knowledge Center is your gateway to product documentation, FAQs, and other sources of Symphony information.

Introducing the consumer

To understand how Symphony supplies resources to meet workload requests, consider this analogy: A bank customer does not withdraw funds directly from the bank vaults—the customer accesses an account, and requests a withdrawal from that account. The bank recognizes the customer by the account number, and determines whether the customer has sufficient funds to make a withdrawal. See Figure 1.

As shown in Figure 2, when a Symphony application requires resources, it does not communicate directly with EGO, and has no direct access to resources. The application is associated with a consumer, and requests resources through it. EGO recognizes the consumer, and through it, allocates resources to the application.



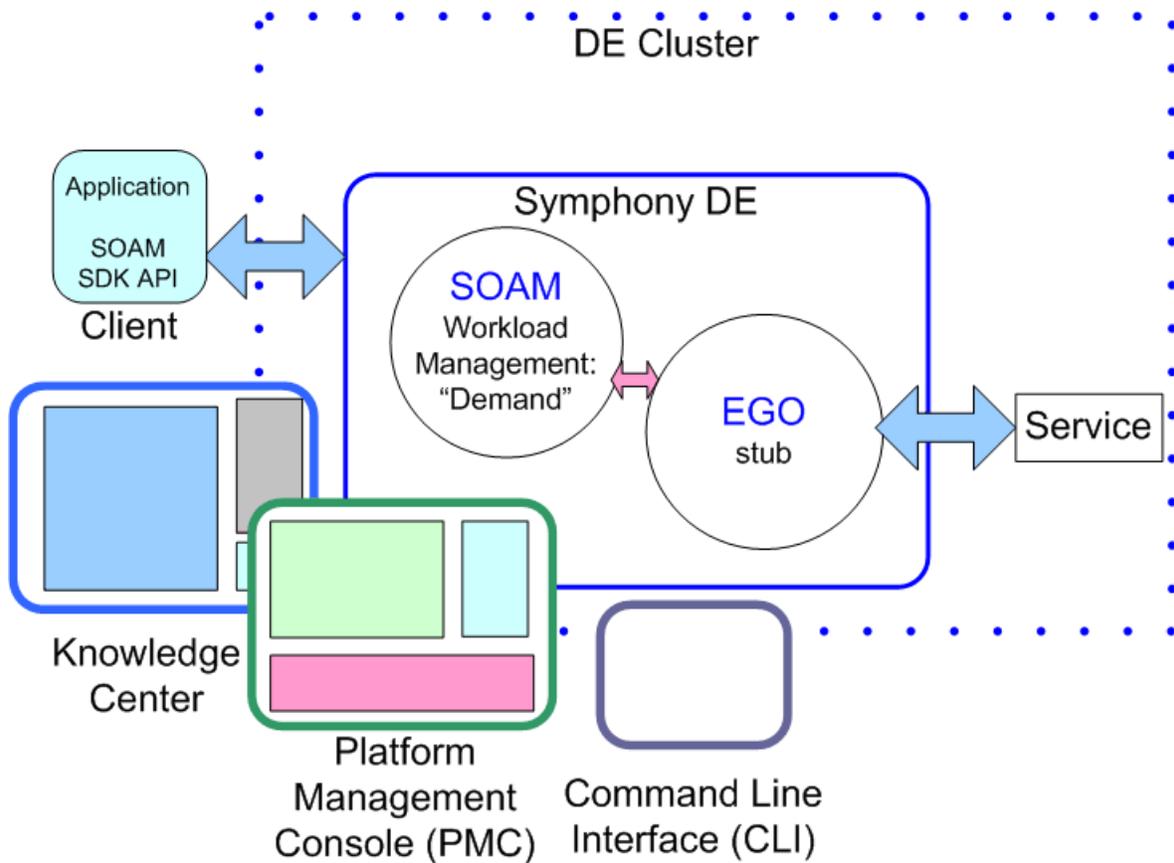
Platform Symphony Developer Edition

Platform Symphony Developer Edition (DE) provides an environment for application developers to grid-enable, test, and run their service-oriented applications. The Symphony DE provides a cluster environment without requiring a full Symphony cluster.

Symphony DE provides a complete test environment, simulating the grid environment provided by Platform Symphony. Developers can test their client and service in their own cluster of machines before deploying to the grid.

Symphony DE provides easy-to-use APIs and rich design patterns to seamlessly grid-enable all types of service-oriented applications with minimal changes. The DE also includes a web interface for monitoring and controlling your test environment, and a knowledge center for easy access to documentation.

To run Symphony workload on the grid, the application developer creates a service package and adds the service executable into the package: no additional code changes are required.

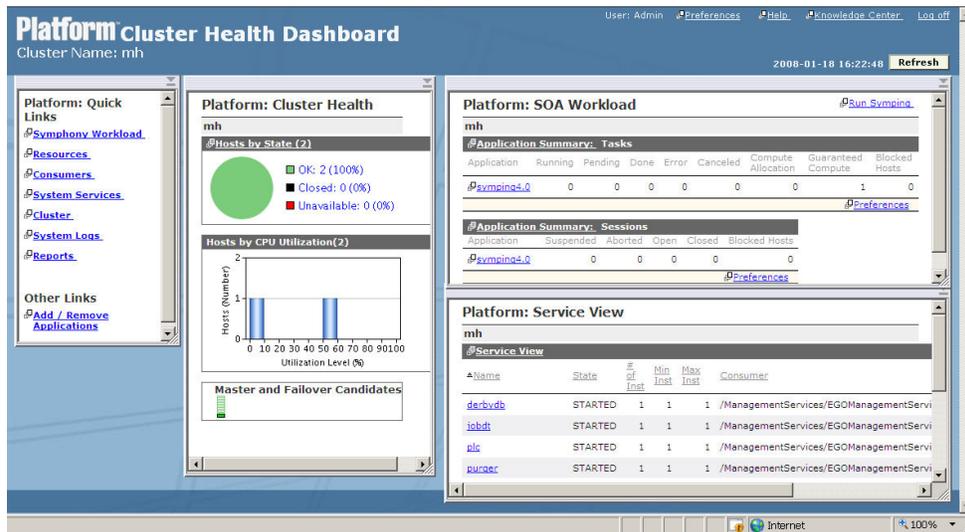


The Symphony DE does not include the EGO resource management component. It does include an EGO stub to simulate basic EGO resource distribution.

Platform Management Console

The Platform Management Console (PMC) is your web interface to Symphony and other Platform products. The PMC provides a single point of access to the key system components for cluster and workload monitoring and control, configuration, and troubleshooting.

Platform Cluster Health Dashboard



The dashboard is the first window you see when you log in to the Platform Management Console. You can access the PMC pages from the dashboard.

The dashboard provides a quick overview of the health of your cluster. It shows a summary of the workload in the cluster, a summary of hosts utilization and status, and links to key pages in the Platform Management Console.

Out of the box, the dashboard looks like the picture above. The dashboard only displays when the console is used to access the grid. It does not appear in Symphony DE.

SOA Workload page

User: Admin [Preferences](#) [Dashboard](#) [Help](#) [Knowledge Center](#) [Log off](#)

Symphony Workload Resources Consumers System Service Cluster System Logs Reports

Monitor Workload Configure Applications Manage Service Packages

cluster1 [Refresh](#)
2007-08-31 11:49:22

Applications [Symping](#) [SubmitWorkload](#)

Application	State	Session summary	Open	Suspended	Closed	Aborted	SSM Host	SSM PID	SI Startup Failures
symexec4.0	enabled	-	0	0	0	0	-	-	-
symping4.0	enabled	-	0	0	0	0	lb01b11.lsf.platform.com	5819	-

[Preferences](#)

The SOA Workload page provides information about any service-oriented workload running in the cluster.

Quick Links page

The Quick Links page provides links to the key pages in the PMC, such as Resources, Consumers, System Logs, Reports and others.

Additional links

The Service View page provides summary information about the services running in the cluster.

In the Symphony DE, the Platform Management Console also provides a link to the Eclipse Development Environment.

Knowledge Center

The Knowledge Center is your portal to Symphony documentation. It is installed with Symphony, and can be accessed from the Start menu on a Windows machine, a URL in any web browser, or a link from the Platform Management Console.

The Knowledge Center provides an overview of the organization of the product documentation. It also provides quick access to each document and links to some key resources such as FAQs, tutorials, and my.Platform.com, your eSupport site.

The Knowledge Center also provides full search capabilities within the documentation. You can perform keyword searches within a specific document, or across the whole documentation set.

The screenshot shows the Platform Symphony Knowledge Center interface. At the top right, there is a link for "Platform Home Page". The main header reads "Platform Knowledge Center". Below this, a blue bar indicates "Platform Symphony 4.0".

The interface is organized into several sections:

- Get Started:** Contains two cards. The first is "Easy Cluster Verification Steps to Get Started" with a "View PDF" link and a description: "A graphical view on steps to take to ensure your cluster is working properly and how to diagnose problems." The second is "Platform Symphony Foundations" with a "View PDF" link and a description: "Introduces you to basic concepts, architecture, and behavior in Symphony."
- Install:** A section header.
- Manage:** Contains four cards: "Cluster and Application Management Guide" (with "View PDF" link), "FAQs" (described as "Frequently asked questions on Symphony and developer topics"), "Symphony Reference" (with "View PDF" link), and "Error Message Reference" (described as "Detailed information about error messages, what generates them, and actions to take, if any").
- Develop:** Contains two cards: "Getting Started: Developer Overview" and "Application Development Guide" (with "View PDF" link).

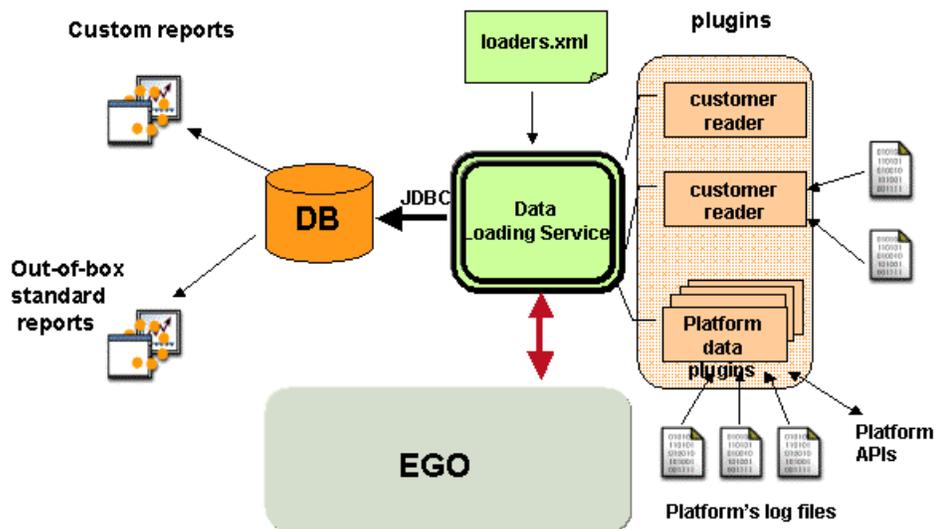
On the right side, there is a search box with a "Search" button. Below it, it states "Wildcards '*' and '?' are supported." and offers search options: "All" (selected), "any words (OR)", and "all words (AND)". The "Results per page" is set to 10. Below the search box is a "Quick Links" section with links for "Cluster commands" (including `eqosh command` and `eqoconfig command`), "Workload commands" (including `soamcontrol command`, `soamdeploy command`, and `soamview command`), "Events and Logs" (including `Logs`, `Events`, `Audit Logs`, and `Traces`), and "Cluster Control" (including `Cluster Management`). At the bottom right, there is a section "On Our Web Site..." with links for "System requirements" and "Send us your feedback".

Reporting

An efficient cluster maximizes the usage of resources while minimizing the average wait time of workload. To ensure your cluster is running efficiently at all times, you can analyze the activity within your cluster to find areas for improvement.

The reporting feature collects data from the cluster and maintains this data in a relational database system. The reporting feature extracts the cluster data from the database and displays this data in reports either graphically or in tables. You can use these reports to analyze and improve the performance of your cluster, to perform capacity planning, and for troubleshooting.

The reporting feature is built on top of the Platform Enterprise Reporting Framework (PERF) architecture. This architecture defines the communication between your cluster, relational database, and data sources.



Symphony collects various types of data, which can be reported on using the standard, out-of-the box reports. In addition, Symphony can be configured to collect customer-specific data, which can be reported on by using custom reports.

Security

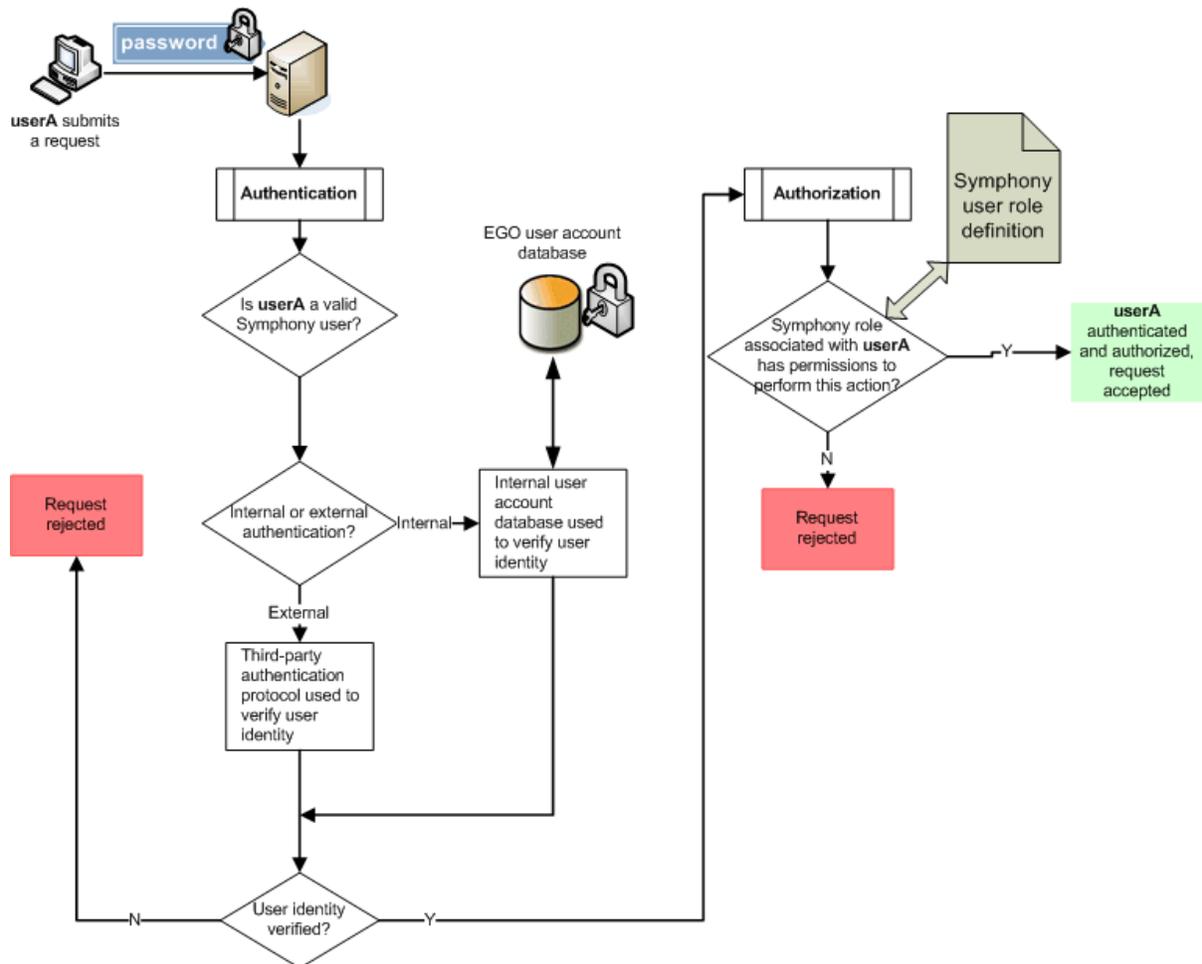
Authentication versus authorization

Authentication is the process of verifying identity. This identity can be a user account used by a person, a host ID used by a machine, a server certificate used by a server software component, or a client certificate used by a client software component. Authentication is usually performed by proving the identity bearer has a secret that is known only to the bearer.

After an identity is authenticated, authorization is the process of determining who is allowed to do what. Authorization is accomplished by assigning privileges or roles to an identity that accesses system objects.

A privilege is an access right to one or more system objects. A role is a collection of privileges. A role can be assigned to one or more users. A user may have multiple roles. Unlike hierarchical users, a role does not contain another role.

Symphony security model



Out of the box, the Symphony security model uses Symphony's own user account database. A user account defined in the database includes a password to provide authentication, and assigned role, which provides authorization.

Symphony also provides a security plug-in for sites that prefer to use a third-party security mechanism, such as ActiveDirectory.

Symphony user roles

Regardless of the authentication method you use, Symphony uses role-based authorization to control access to system objects. Symphony supports the following default roles:

- The Cluster Administrator role can administer any objects and workload in the cluster
- The Consumer Administrator role can administer any objects and workload in consumers to which they have access
- The Consumer User role can run workload in consumers to which they have access

Symphony also offers administrators the ability to choose from a set of predefined permissions and apply them to new or existing user roles.

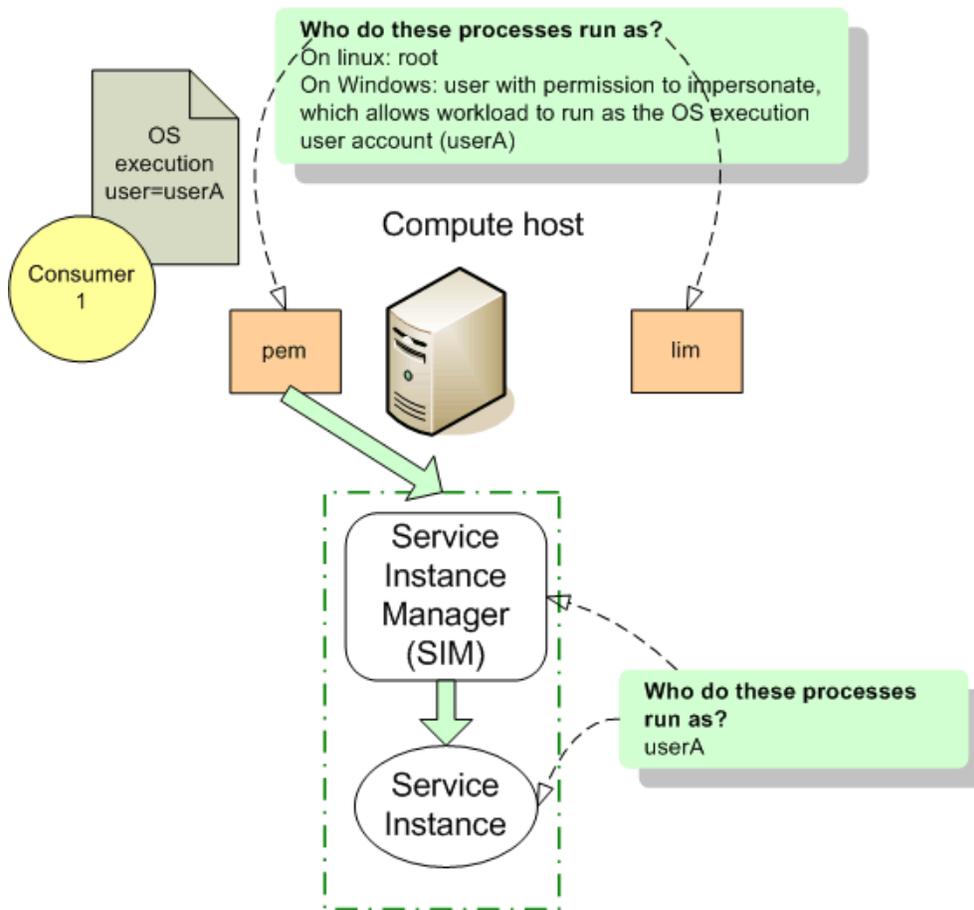
User accounts are created and managed in EGO. EGO authorizes users from its user database.

Each consumer is associated with a list of user accounts that are allowed to access the consumer. Different user accounts can submit or control workload. However, each consumer is associated with only one user account for running workload—all workload that runs under one consumer runs under one operating system account.

Impersonation

Sometimes service instances require user-specific privileges to access certain resources. Sometimes due to a security policy at an organization, it is necessary to isolate the user under which a process runs. Symphony provides flexible ways to configure the user account under which workload runs, allowing you to isolate users and applications.

Impersonation means that the system runs executables under a designated operating system account.



Security across communication channels

Symphony uses open Secure Socket Layer (SSL) to secure communications between components. SSL is a protocol that uses encryption and authentication techniques to secure connections between clients and servers.

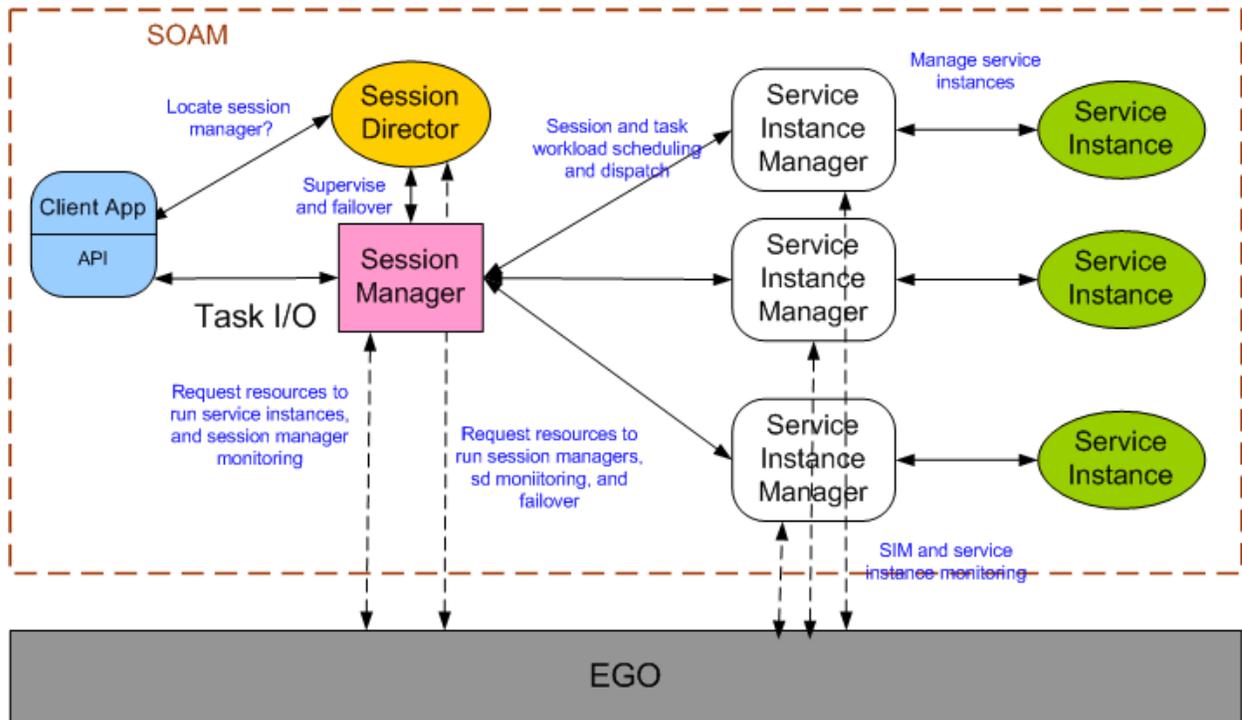
2

Inside Workload Management

SOA Middleware components

The Symphony service-oriented middleware components consist of the Session Director, session manager (s), service instance managers, and service instances.

The following diagram shows the components and interactions between them.



Session Director

There is one Session Director in a Symphony cluster. It is started as an EGO system service.

The Session Director provides a single entry point to the SOA middleware, and does the following:

- Authenticates the connecting client
- Manages the session manager lifetimes, and acquires and provisions resources for the session managers, as required
- Provides a web services interface for administrative operations

Session manager

There may be a single session manager, or multiple session managers in a Symphony cluster. The number of session managers depends on the number of running applications. A session manager is created upon the first user connection to an application for which no session manager exists, or it may be configured to prestart even if there is no workload.

A session manager does the following:

- Routes task input and output messages between the client application and compute hosts
- Obtains resources from EGO to service its sessions

- Schedules sessions based on various scheduling policies (proportional, minimum service, and so on)
- Provides administration and control operations
- Logs status and progress of sessions and tasks
- Provides fault-tolerant message handling as specified by the application profile
- Manages the life cycle of service instance managers, and manages the interaction with the resource manager

Service instance manager

There is one service instance manager for every instantiation of a service. A service instance manager is created and destroyed by the session manager.

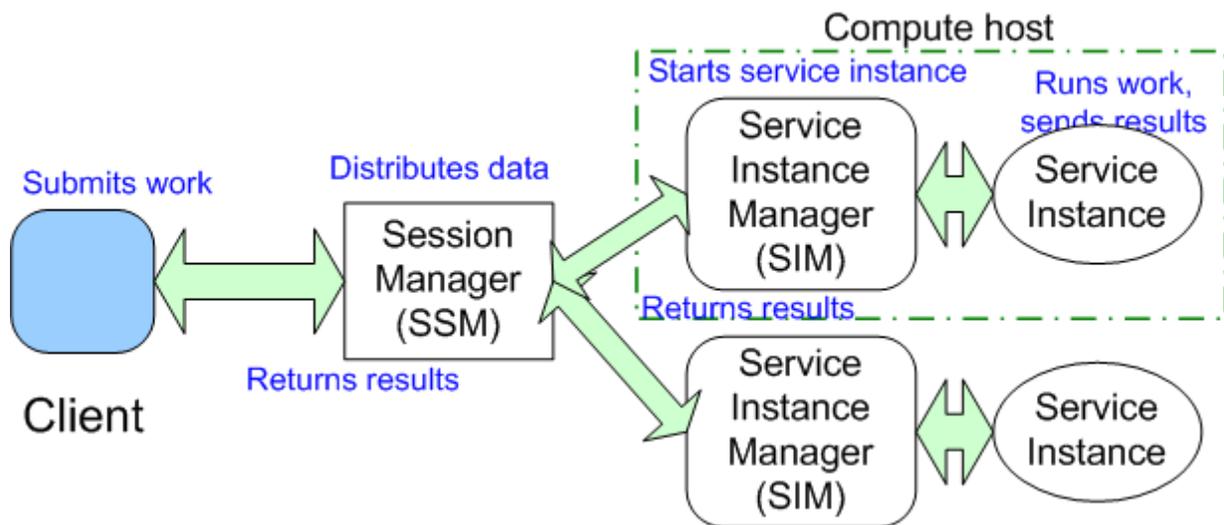
A service instance manager (sometimes referred to as a SIM), is responsible for the following:

- Starting and managing the life cycle of a service instance, including monitoring the health of the service instance
- Routing messages between the session manager and a service instance

Service instance

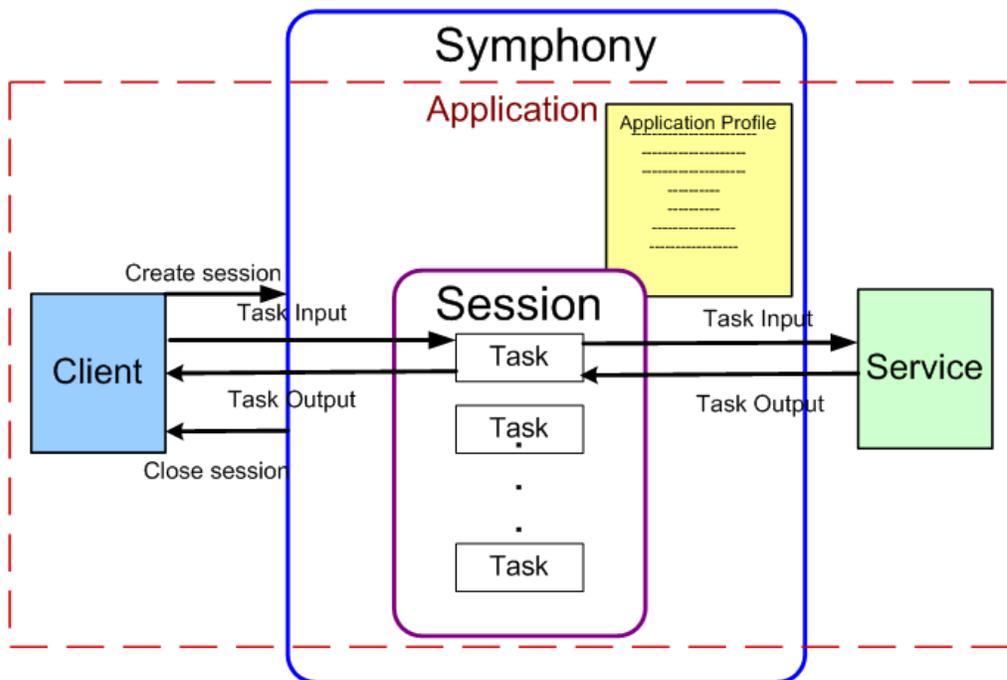
A service instance is a single instantiation of the service that executes when the application runs. There can be many instances of the same service running at any given time.

Data flow



Service-oriented application objects

Symphony service-oriented applications consist of a client application, and a service. When the application runs, a session is created, containing a group of tasks. The application profile provides information about the application.



Application profile

The application profile defines characteristics of the application, and defines the behavior of the middleware and the service. There is one application profile per application.

Session

A session is created via the Symphony client API. A session consists of a group of tasks that share common characteristics, such as data required for computations. Each session has a session ID that is generated by the system. Session IDs are unique within an application.

Task

A task is an autonomic computation unit, and is the basic unit of work in a service-oriented application. Tasks perform computations in parallel. Each task has a task ID, unique within the session.

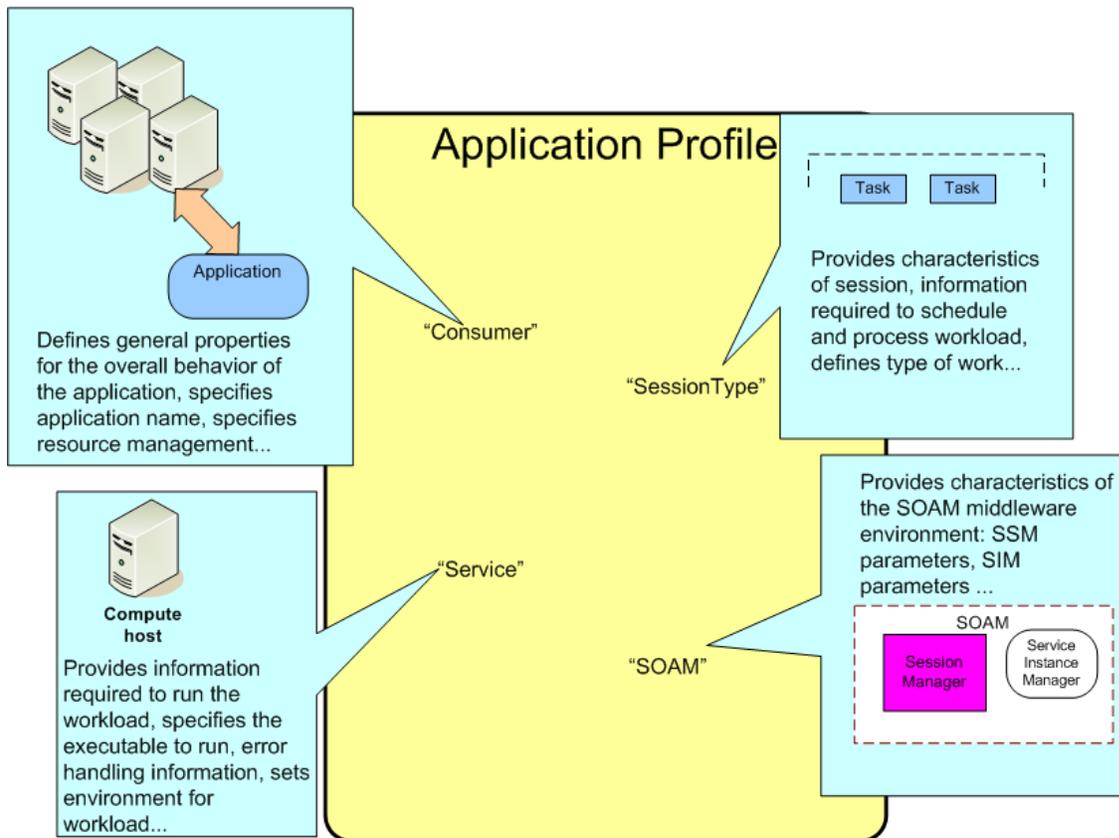
Each task can have an input message and an output message. The input message is the input to the computation, the output message is the result of the computation.

Application profile

The application profile defines the characteristics of an application and the environment in which the application runs.

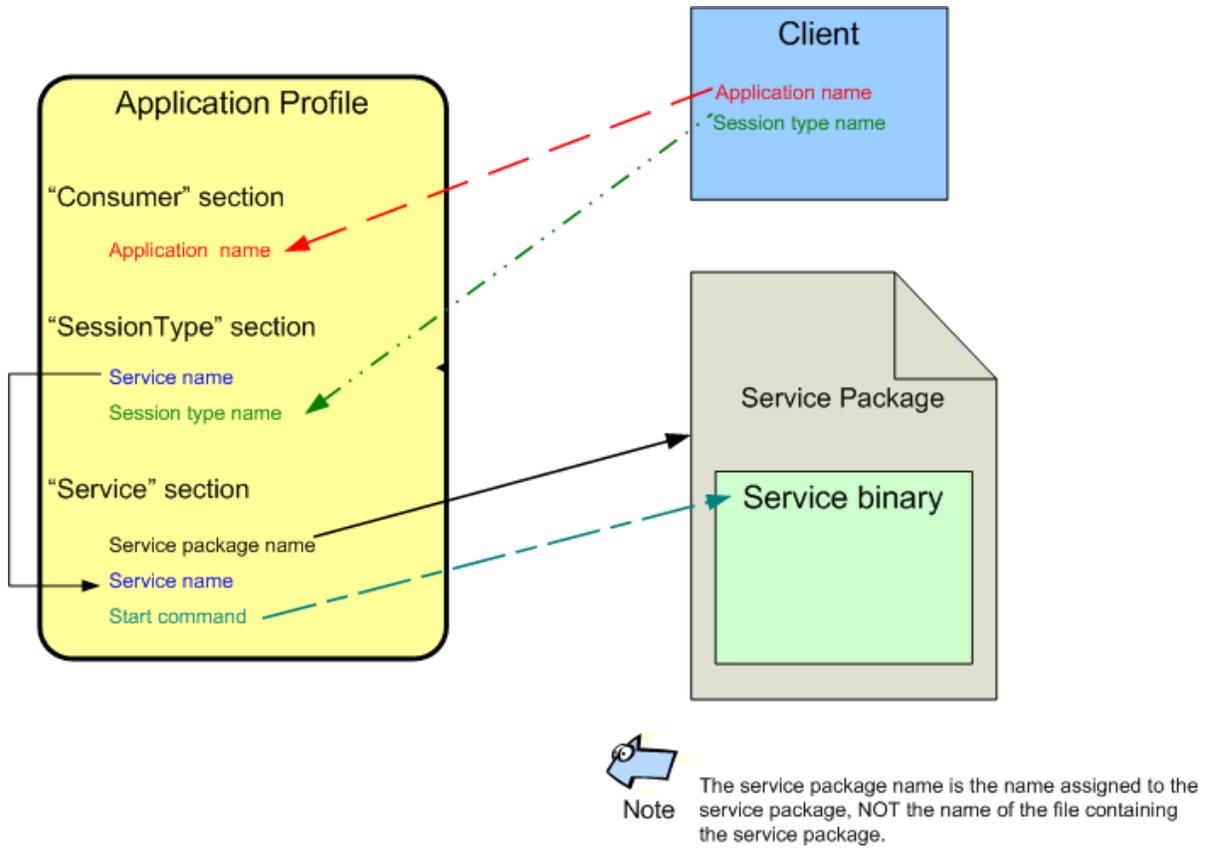
The application profile provides the following information:

- The information required to run the application
- The scheduling policies that apply to the application
- Configuration information for the session manager and the service instance managers
- Configuration information for sessions and services



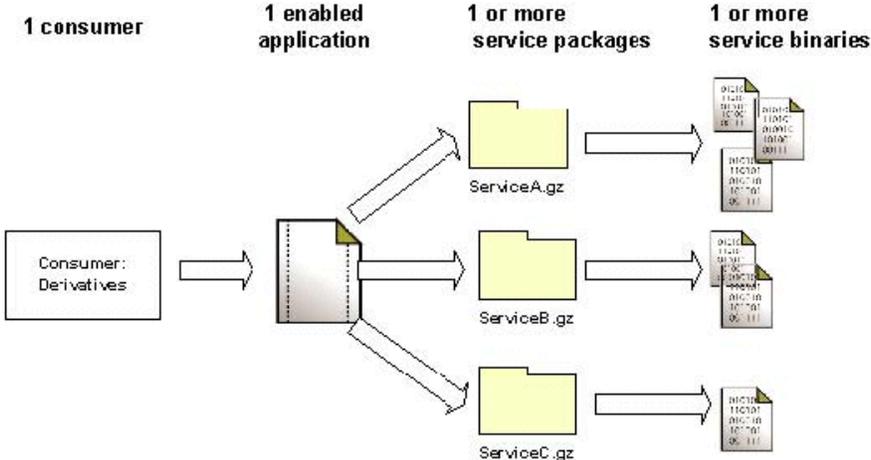
Relationship to the client, service package, and service

The application profile provides the linkage between the application, the client, the service package, and the service. The following illustrates how parameters in the application profile relate to the client, service and service package, and which values must match.



Consumers, applications, services, and service binaries

The following illustrates the relationships among consumers, applications, services, and service binaries. For every consumer, there is one enabled application. One or more service packages can be deployed for an application. A service package contains the binary program that performs computations. A service package can contain more than one service binary, for example, Linux and Windows versions of a service.



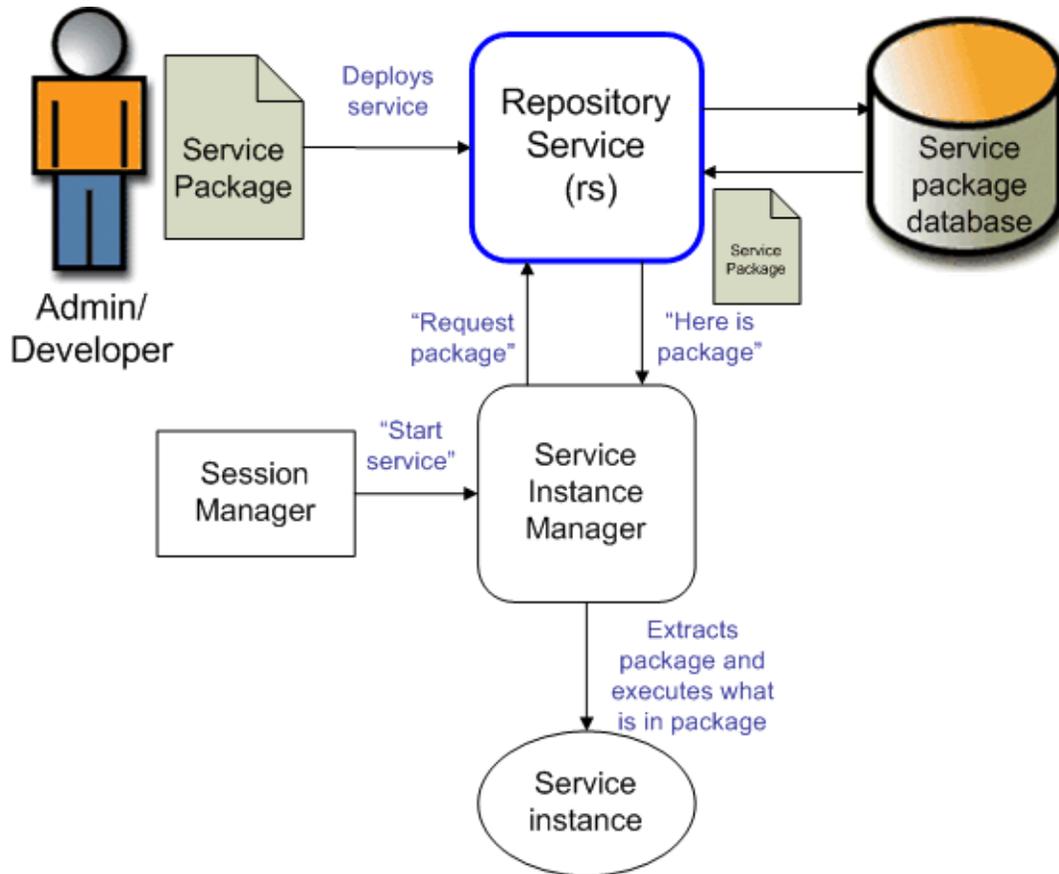
Service package deployment

Symphony services are deployed to the cluster and made available in either of the following ways:

- Using the Symphony repository service
- Using a third-party deployment tool

Deployment using the repository service

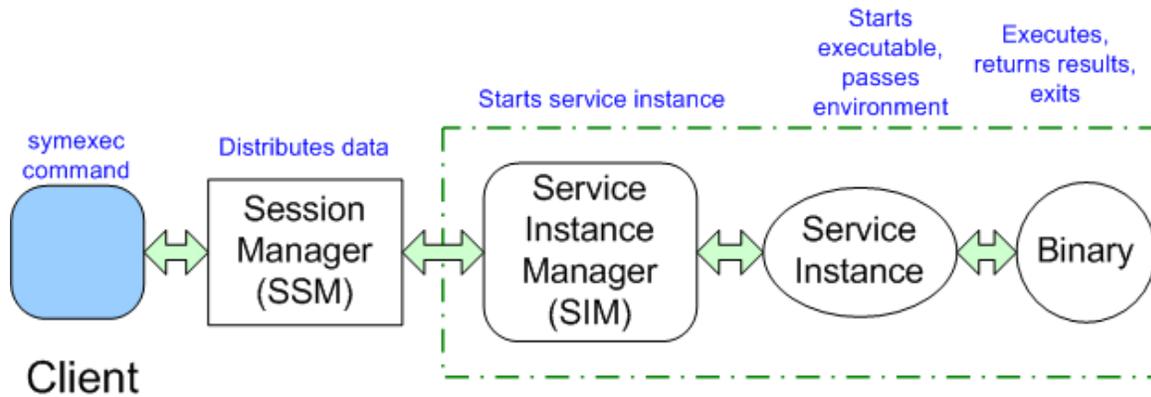
An administrator or developer deploys a service package to the repository service. When a compute host needs the package, it requests the package from the repository service.



Running executables in Symphony

Symphony provides the ability to run existing executables as Symphony workload without code changes. Executables are handled much the same as SOA workload, with the exception of the following:

- A specialized service instance runs the executable
- The specialized service instance starts, runs the executable and exits when the executable finishes



C H A P T E R

3

Inside Resource Management

EGO component overview

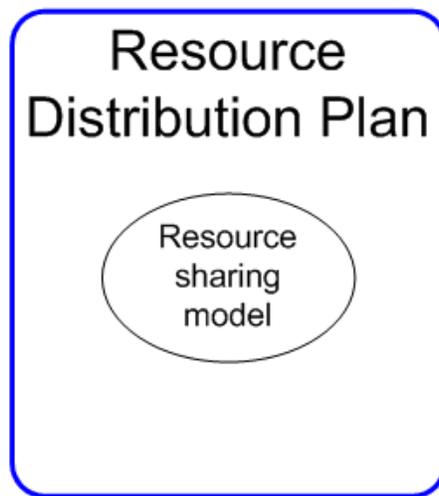
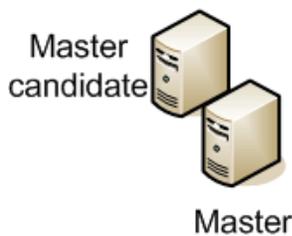
EGO provides the underlying system infrastructure to control and manage cluster resources.

EGO provides the underlying system infrastructure to enable multiple applications to operate within a shared resource infrastructure in Symphony.

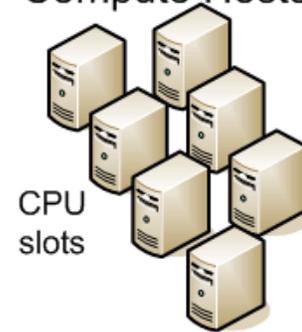
Just as an operating system running on a single machine aggregates and virtualizes physical resources and allocates them to applications, EGO performs similar functions, but across a distributed environment.

EGO manages both logical and physical resources and supports all forms of applications. EGO manages the supply of resources, making them available to applications.

Management Hosts



Compute Hosts



Hosts can be divided into two groups: management hosts and compute hosts. Management hosts provide specialized services to the cluster, while compute hosts run user workload.

Management hosts

Management hosts provide both cluster and workload management services within the cluster, and are not expected to run workload for users. The master host, all master candidate hosts, and session manager hosts must be management hosts. Other management hosts include the Web server host and the host running the data loaders and data purger for the reporting feature.

Management hosts all run on the same operating system: all Windows or all UNIX.

- Master host** The master host is the first host installed in the cluster. The resource manager (vemkd) for the cluster resides on this host. The master host controls the rest of the hosts in the cluster and is the interface to the clients of the cluster.
- Master candidates** There is only one master host at a time. If the master host should fail, another host automatically takes over the master host role. Hosts that can act as the master are called master candidates.

Session manager host	One or more management hosts run session managers. There is one session manager per available slot on a management host. There is one session manager per application.
Web server host	The Web server host runs the Platform Management Console. Only one management host is elected as the Web server host.

Compute hosts

Compute hosts are those hosts in the cluster that provide computing resources to consumers. A cluster may contain any number of compute hosts, but must have at least one compute host.

CPU slots A CPU slot is the unit used to measure compute resources. A single CPU slot can run one service instance on a compute host, or one session manager on a management host.

Resource distribution plan

The resource distribution plan implements a resource sharing model for the cluster. The resource sharing model defines how cluster resources are distributed among applications. The resource sharing model specifies whether resources are reserved for exclusive use by an application, shared between specific applications, or pooled together and shared based on a ratio. There are several models to choose from.

Daemons

- **VEMKD:** The VEM kernel daemon that runs on the master host. It starts other daemons and responds to allocation requests
- **EGOSC:** The EGO service controller requests appropriate resources from the VEMKD and controls system service instances.
- **PEM:** Process execution manager works for the VEMKD, starting, controlling, and monitoring activities, as well as collecting and sending run time resource usage.

Resources

Resources are physical and logical entities that are used by applications in order to run. While resource is a generic term, and can include low-level things such as shared memory segments or semaphores, in Symphony, EGO manages CPU slots.

A resource of a particular type has attributes. For example, a compute host has the attributes of memory, CPU utilization, operating system type, and so on.

Resource groups

Resources may be grouped together into logical groups to simplify identification, resource allocation, or for administration and monitoring purposes. These resource groups are used to provide a consumer with a like group of hosts to run workload—any host in a resource group should be able to run the same workload.

Figure 1:
Out-of-Box Resource Groups

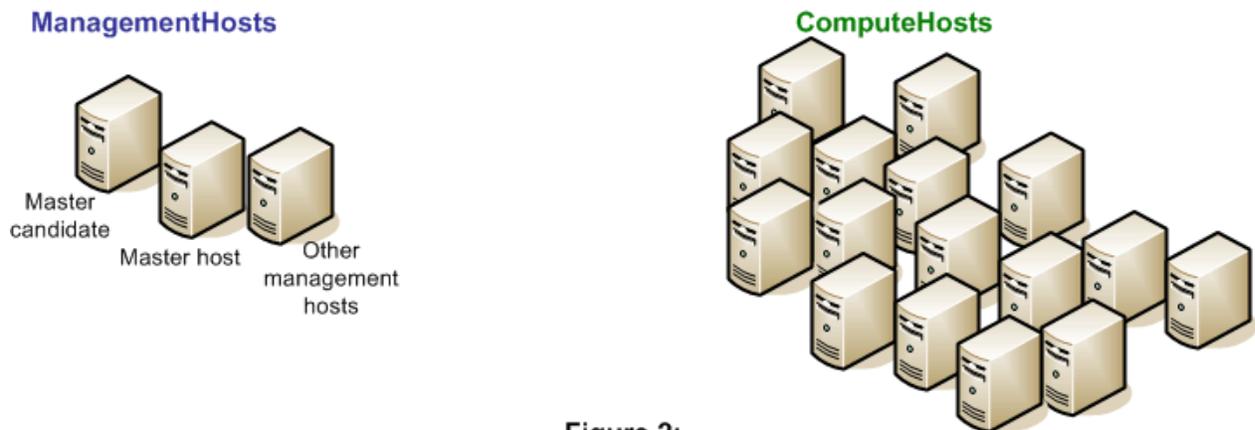
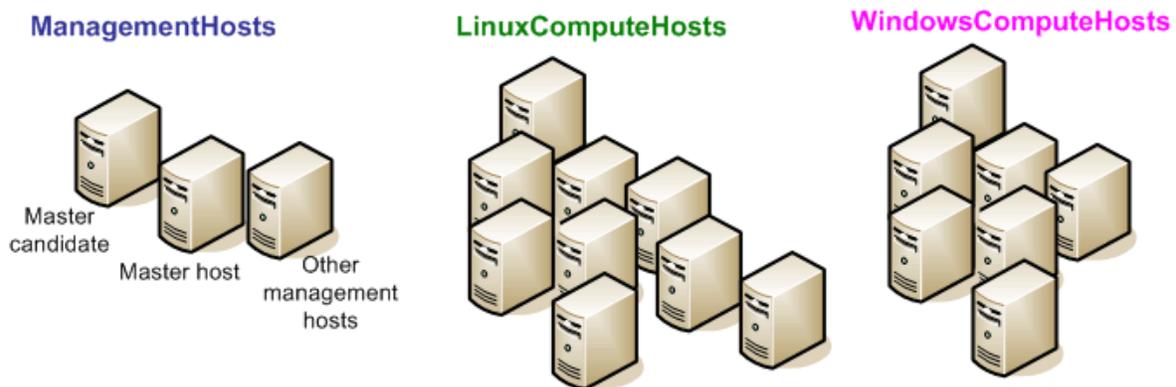


Figure 2:
Grouping Like Resources



As shown in Figure 1, there are two resource groups out of the box:

- ManagementHosts
- ComputeHosts

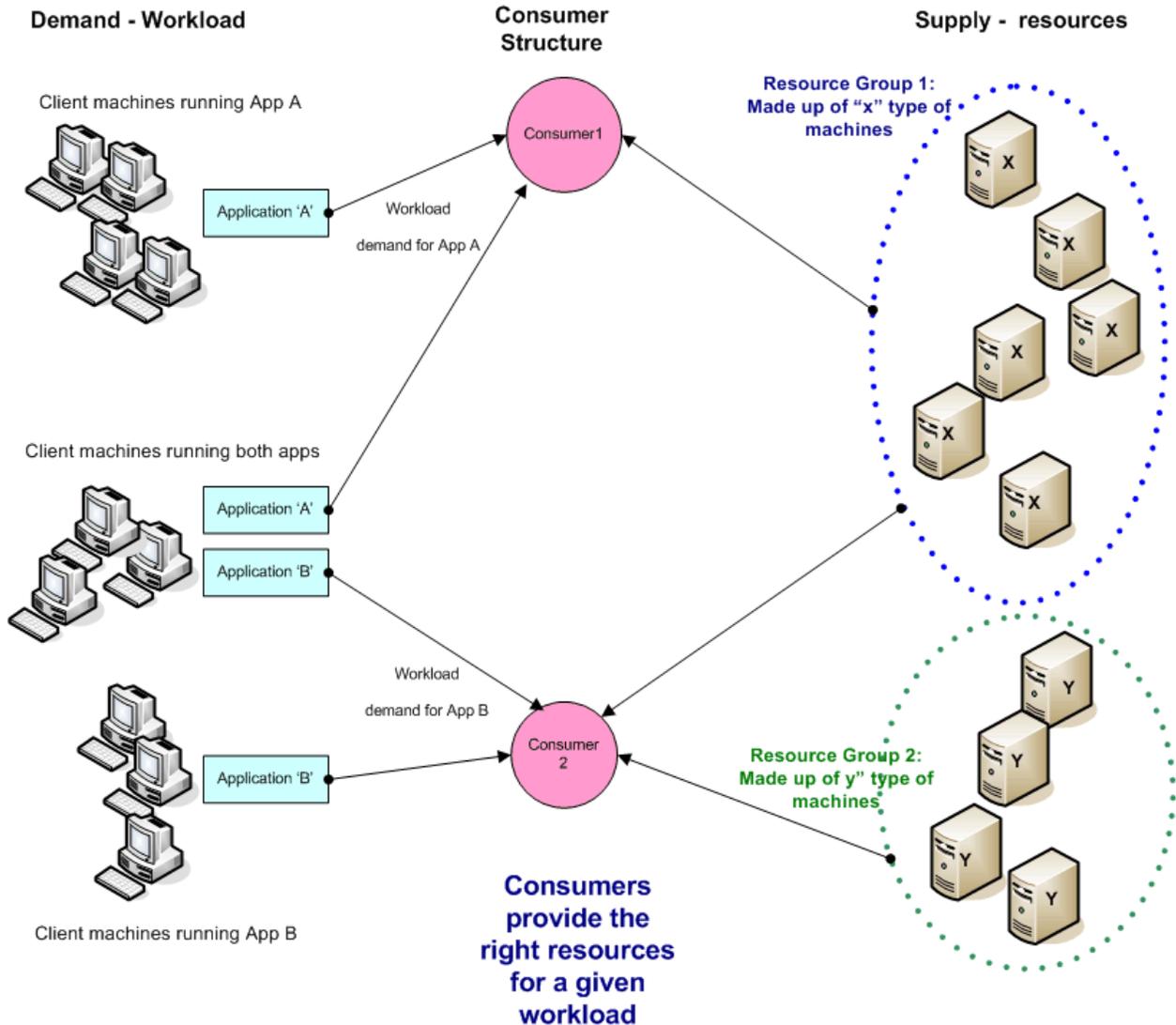
If all of your hosts are identical, these resource groups may suffice. If your application requires a specific type of hosts (for example, with a minimum processor speed), and not all hosts meet this criteria, you likely need to create resource groups to group like hosts together.

For example, a simple way to group resources may be to group your hosts by operating system type.

EGO provides a common grouping mechanism for resources. Resources may come and go from the system, so EGO supports dynamic membership in a resource group. Hosts can be placed explicitly into individual resource groups, or the resource groups can be defined to have a dynamic membership based on specific criteria. This criteria includes operating system type, CPU speed, total memory, or swap configuration, or custom attributes.

Consumers

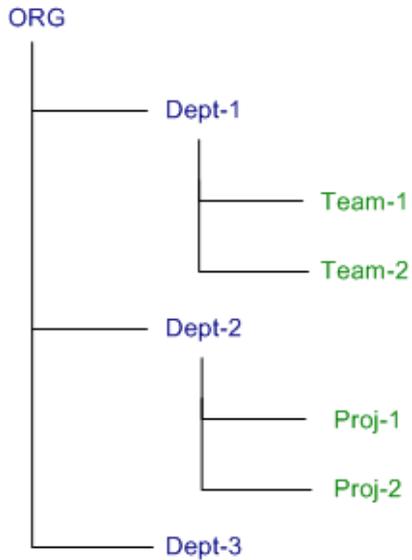
A consumer is a unit within the representation of an organizational structure. The structure creates the association between the workload demand and the resource supply.



Consumer structure

The consumer structure is hierarchical, and represents some logical, organizational structure. The consumer structure may mirror the organizational structure of a business unit, where each consumer represents a department. The consumer structure can represent any type of organization, provided that the structure represents the organizations that want to access compute resources.

A consumer can be an individual user, a project, a department, or an entire company.



A consumer can be divided into lower-level consumers, which may also be subdivided. The lowest-level consumer is the level at which an application is associated.

In the above example, both Dept-1 and Team-1 are consumers. Team-1 is a child consumer of Dept-1. However, Team-1 is a leaf consumer—the lowest level consumer in its hierarchy.

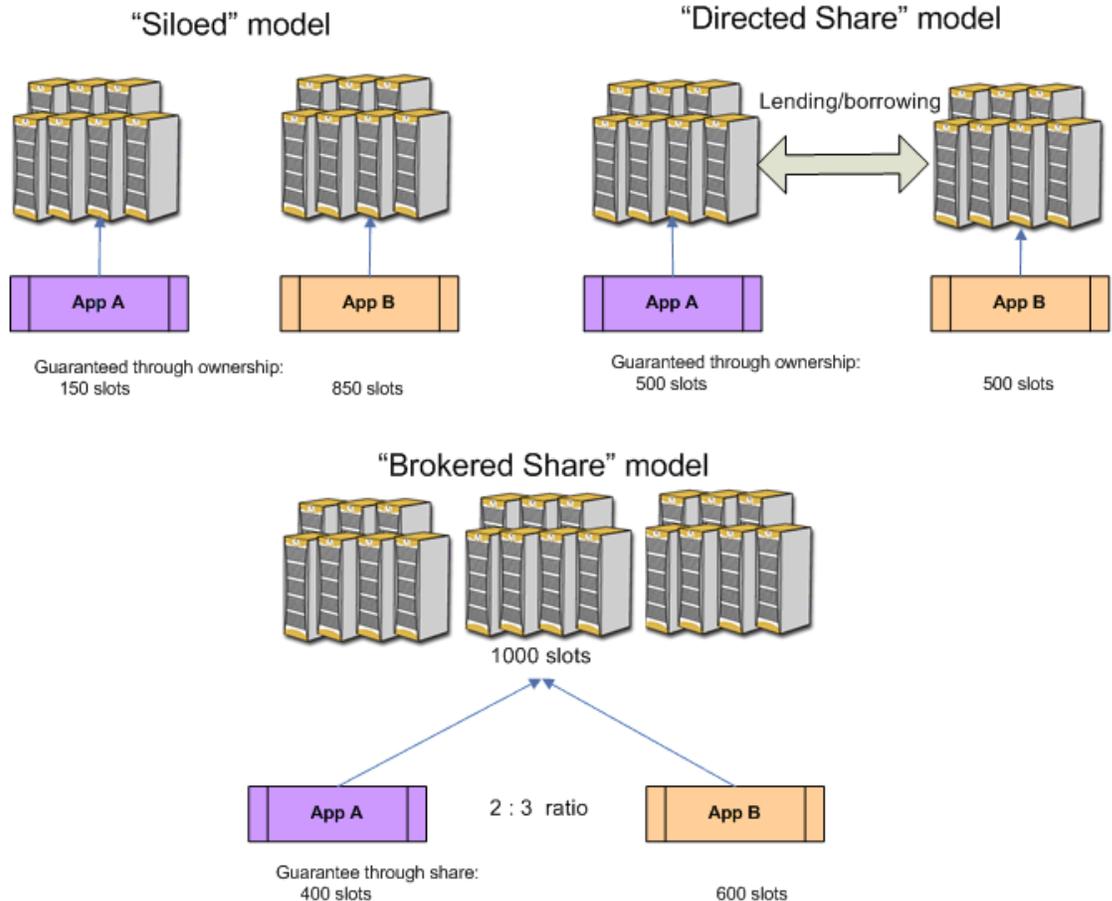
A consumer is identified by a string that represents a path in the hierarchy. Each element of the path represents a consumer. The administrator at each level in the consumer hierarchy can define its child consumers, and the policies that determine how the consumer's resources are shared between the child consumers.

Resource sharing models

Resources are distributed in the cluster as defined in the resource distribution plan. The resource distribution plan may implement one or more resource sharing models.

There are three key resource sharing models:

- Siloed model
- Directed share model
- Brokered share or utility model



Siloed model

The siloed model guarantees resource availability to all consumers. It is strictly a 'siloed' ownership model: consumers do not share resources, nor are the cluster resources pooled. Each application brings its designated resources to the cluster, and continues to use them exclusively.

In the above example, in a cluster with 1000 slots available, application A has exclusive use of 150 CPU slots, and application B has exclusive use of 850 slots.

Directed share model

The directed share model is based on the siloed model: consumers own a specified number of resources, and are still guaranteed that number when they have demand. The directed share model allows a consumer to lend its unused resources to sibling consumers when their demand exceeds their owned slots.

In the above example, applications A and B each own 500 slots. If application A is not using all of its slots, and application B has a requirement for more than its owned slots, application B can borrow a limited number of slots from application A.

Brokered share or utility model

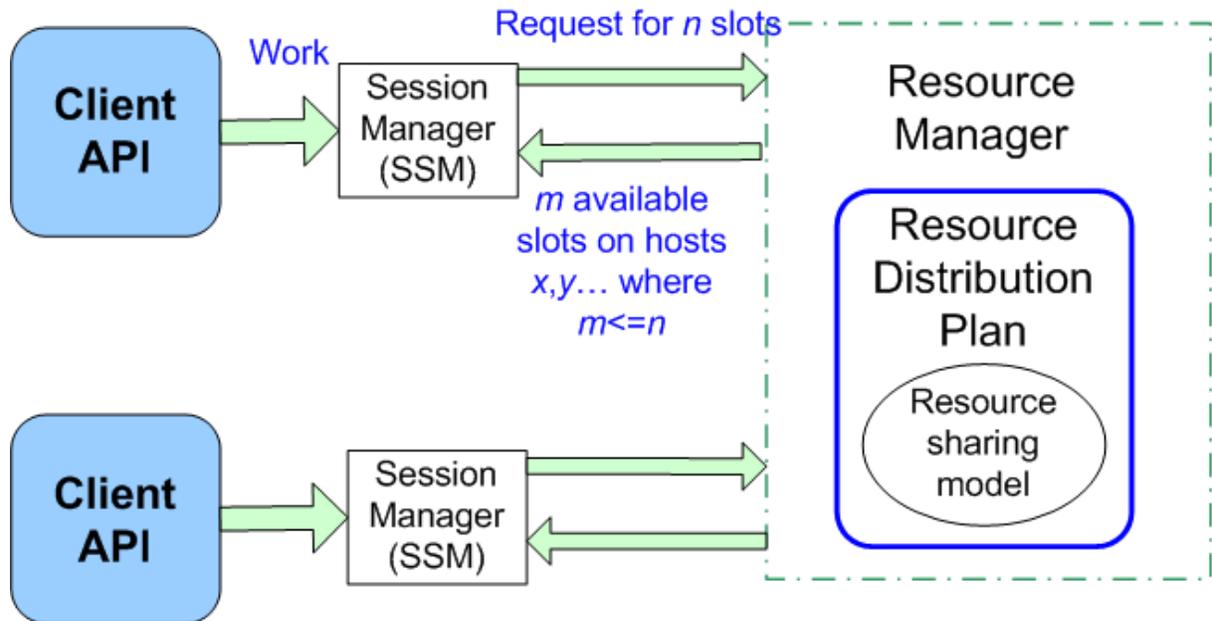
The brokered share or utility model is based entirely on sharing of the cluster resources. Each consumer is assigned a proportional quantity of the slots in the cluster. The proportion is specified as a ratio.

In the above example, application A is guaranteed two of every five slots, and application B is guaranteed three. Slots are only allocated when the demand exists. If application A has no demand, application B may use all slots until application A requires some.

Sharing of Symphony resources

Symphony resources are shared as defined in the resource distribution plan.

A client makes a request of the session manager, which requests a certain number (n) slots of the resource manager. Based on the values specified in the resource distribution plan, the resource manager returns the number of available slots (m) and the names of the hosts on which the slots reside.



Resource distribution plan

Resources are distributed based on the relationships and values specified in the resource distribution plan.

A *policy* is a set of rules that defines a behavior for scheduling or resource distribution.

The *resource distribution plan* is a collection of distribution policies that describes how EGO supplies resources to satisfy the workload demand.

Resource distribution policies

Resource distribution policies define how many resources each application may use, whether the consumer has sole ownership of the resources, or shares them with other consumers. If resources are shared, resource distribution policies define how they are shared, who they are shared with, and when. The resource distribution policies also define how resources are reclaimed when lent, and how they are disbursed when there is contention for the resources.

The following resource distribution policies may apply:

- Ownership policy
- Sharing policy
- Borrowing/lending policy
- Reclamation policy
- Rank

Ownership policy	Defines guaranteed access to a set number of resources. The consumer receives the guaranteed number of resources regardless of its outstanding demand.
Sharing policy	<p>Defines how unowned resources are shared amongst consumers. Any resource that is not explicitly owned is placed in a shared pool. These resources are shared based on a ratio specified in the sharing policy. Shared resources are allocated as follows:</p> <ol style="list-style-type: none"> 1. Consumers receive resources based on their <i>share ratio</i> (a ratio defining a relative portion of the resources), up to a maximum value, their <i>deserved</i> value. 2. Any free resources remaining in the shared pool are distributed among consumers based on their share ratio. Share ratio allows more important sibling consumers to borrow more resources than less important siblings.
Borrowing/ lending policy	Defines a relationship between consumers such that unused resources belonging to a consumer may be lent to a sibling consumer. The policy defines limits on the borrowing: a policy may limit borrowing to specific consumers, and limits may be applied to the number of resources that a consumer may borrow. Borrowing order is configurable, to allow borrowers with the highest rank to be processed first.
Reclamation policy	Defines the criteria under which the lender reclaims its owned resources from borrowers. The policy can specify to allow a grace period before initiating the resource reclaim, or the policy can specify to kill any running workload and reclaim the resources immediately.
Rank	Rank applies to the order in which other policies are applied to consumers. Rank determines the order in which the distribution of resources is processed—the highest ranking consumer receives its resources first, borrows resources first, and returns borrowed resources last.

C H A P T E R

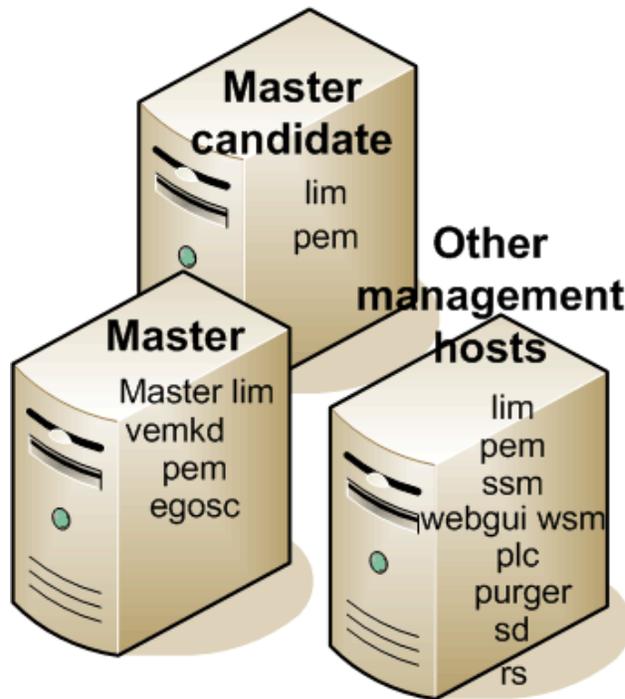
4

Inside the Symphony Cluster

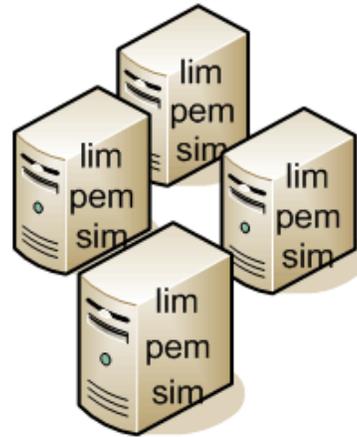
Symphony processes

There are multiple Symphony processes running on each host in the cluster. The type and number of processes running depends on whether the host is a management host or a compute host.

Management Hosts



Compute Hosts



Management host processes

Symphony management hosts run various processes, depending on their role in the cluster.

On the master host

- master** The master `lim` starts `vemkd` and `pem` on the master host. There is one master `lim` per cluster.
- lim**
- vemkd** The `vemkd` (EGO kernel) does the following:
 - Starts the service controller `egosc`
 - Maintains security policies, allowing only authorized access
 - Maintains resource allocation policies, distributing resources accordingly
 - Serves as an information center where clients can query information about the clusterThere is one `vemkd` per cluster, and it runs on the master host.
- pem** The `pem` (process execution monitor) monitors `vemkd`, and notifies the master `lim` if `vemkd` fails.

egosc The `egosc` (EGO 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. There is one `egosc` per cluster, and it runs on the master host.

On other management hosts

sd The `sd` (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 other management host.

rs The `rs` (repository service) provides a deployment mechanism for service packages to the compute hosts in the cluster. There is one repository service per cluster, and it runs on the master or other management host.

lim The `lim` (load information manager) monitors the load on the management host, and starts `pem`.

pem The `pem` (process execution manager) starts Symphony processes on the host.

ssm The `ssm` (session manager) is the primary workload scheduler for an application. There is one session manager per application.

wsm The `wsm` (web interface service) runs the Platform Management Console.

plc The `plc` (PERF loader controller) loads data into the reporting database.

purger The `purger` (PERF data purger) purges reporting database records.

On the master candidate hosts

lim The `lim` (load information manager) monitors the load on the master candidate host. `lim` also monitors the status of the master `lim`. If the master host fails, `lim` also elects a new master host.

pem The `pem` (process execution manager) starts Symphony processes on the host.

Compute host processes

lim The `lim` (load information manager) monitors the load on a compute host, and passes the load information to the master `lim` on the master host.

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

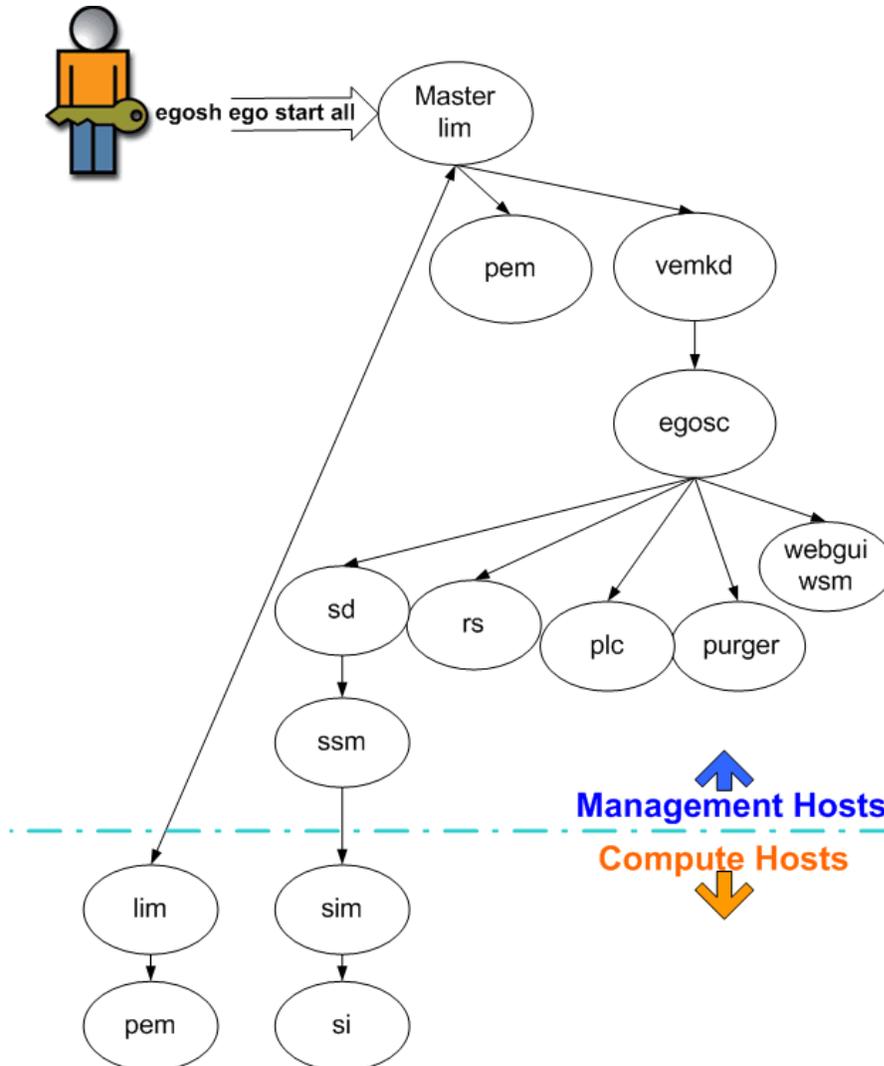
- Starts `pem` on that host
- Provides system configuration information to the master `lim`
- Monitors load and provides load information statistics to `vemkd` and users

pem The `pem` (process execution manager) monitors the `lim` process.

sim The `sim` (service instance manager) is started on the compute host when workload is to be submitted to the host if the application is preconfigured. The `sim` then starts the service instance running. There is one service instance manager per service instance.

Symphony cluster startup process

When the administrator starts the cluster using the `egosh ego start all` command, the processes in the cluster are started as shown below:



1. The `lim` is started on the master host.
2. The master `lim` starts both `pem` and `vemkd` on the master host, and `lim` on all compute hosts.
3. `vemkd` starts `egosc`.
4. `egosc` starts `sd` (session director), `rs`, `plc`, `purger`, and `wsm` using `pem`.
5. `sd` starts `ssm`.
6. `lim` on each compute host starts `pem` on the same host, and reports information to the master `lim`.
7. `ssm` starts `sim` on a compute host (upon a request for resources from a client).
8. `sim` starts `si`.

Fault tolerance for workload

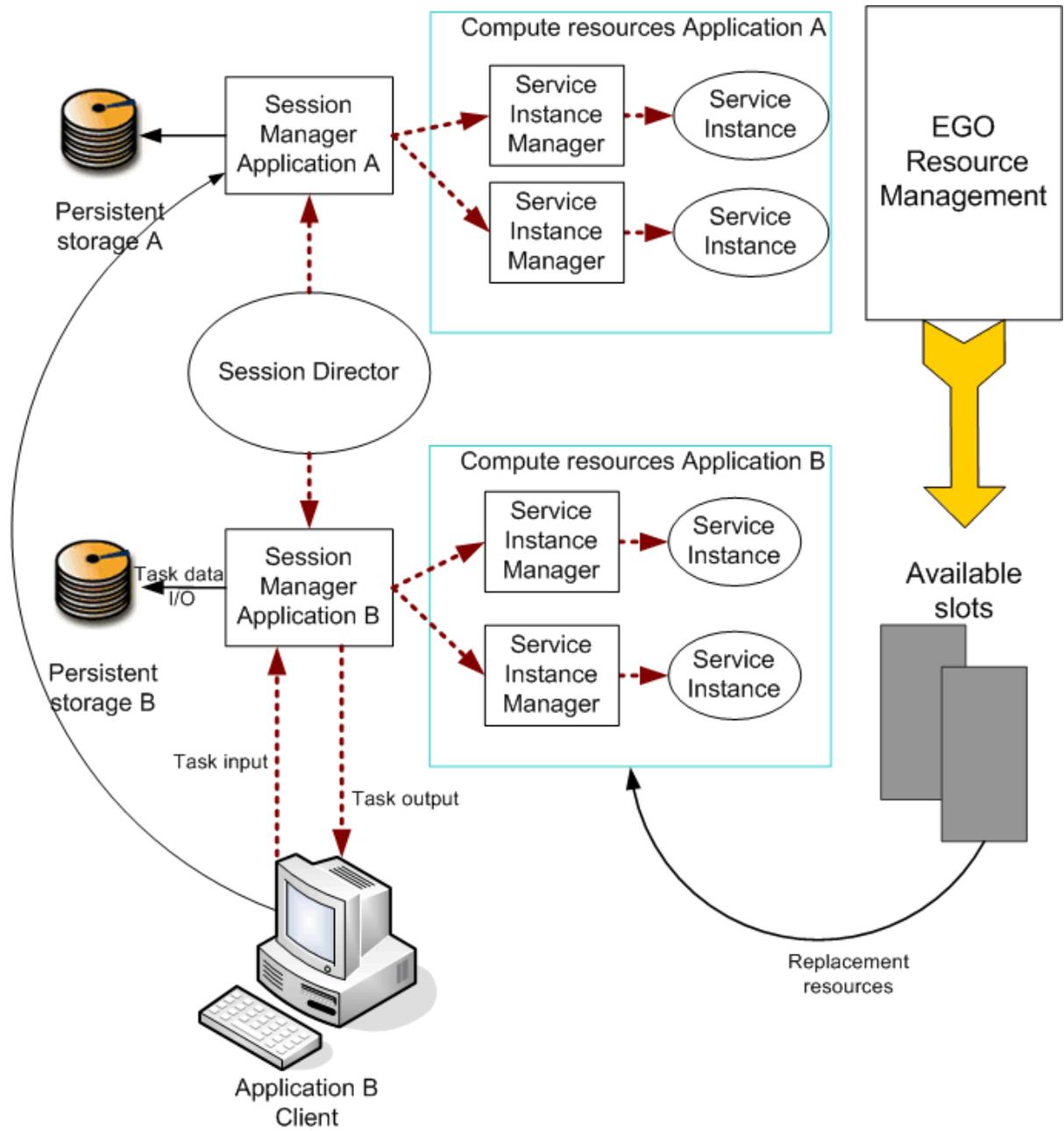
Symphony provides the following fault tolerance for workload:

- The service instance is monitored and recovered by its service instance manager
- The service instance manager is monitored and recovered by its session manager together with EGO
- The session manager is monitored and recovered by session director together with EGO
- Workload (sessions and tasks) can be specified as recoverable or not
- Task delivery is guaranteed

Fault tolerance among applications

Symphony provides the following fault tolerance for applications:

- All service-oriented application properties are defined in application profiles throughout the cluster
- Administrators are in full control of Symphony applications for their assigned profiles
- If EGO encounters an issue, it does not affect running Symphony applications
- If one service instance fails, it does not affect any others
- Persistence, history logging, and data cache are managed per application

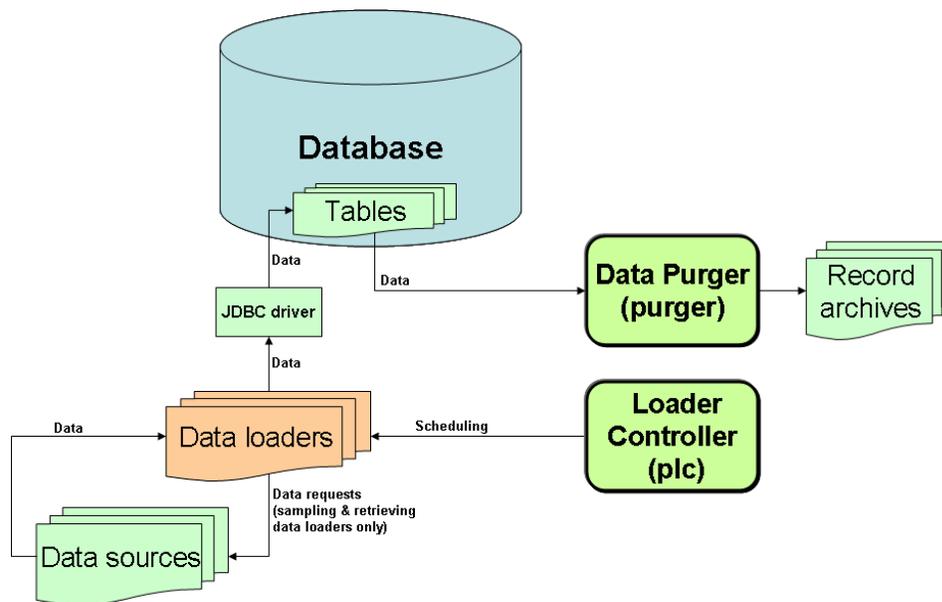


Inside PERF

Platform Enterprise Reporting Framework (PERF) provides the infrastructure for the reporting feature.

Note:

Reporting is only available if you have Platform Symphony or Platform LSF with EGO enabled installed.



Database

Platform Symphony includes the Apache Derby database, a JDBC-based relational database system, for use with the reporting feature. The Derby database is a small-footprint, open-source database, and is only appropriate for demo clusters. If you want to use the reporting feature to produce regular reports for a production cluster, you must use a supported commercial database.

Data sources

Data sources are files that store cluster operation and workload information such as host status changes, session, and task status, and so on. Symphony uses several files as data sources. These include daemon status files, and event files.

Data loaders

Data loaders collect the operational data from the data sources and load the data into tables in a relational database. The data loaders connect to the database using a JDBC driver. The data loaders handle daylight savings automatically by using GMT time when collecting data.

Loader controller

The loader controller service (plc) controls the data loaders that collect data from the system and writes the data into the database.

Data purger

The data purger service (purger) maintains the size of the database by purging old records from the database and archiving them. By default, the data purger purges all data that is older than 14 days, and purges data every day at 12:30am.

Reports

Platform provides a set of out-of-box report templates, called standard reports. These report templates allow you to produce a report to analyze your cluster. The standard reports capture the most common and useful data to analyze your cluster.

You may also create custom reports from advanced queries to capture data beyond what is displayed in the standard reports.

Index

A

- application profile
 - description 23
 - overview 22
 - relationship
 - to client 23
 - to service 23
 - to service package 23
- applications
 - relationship to service packages 25
- authentication
 - definition of 16
- authorization
 - definition of 16

B

- binaries
 - running on Symphony 27
- borrowing policy 39
- brokered share resource model 37

C

- client
 - description of 7
- cluster
 - components of 8
 - description of 6
 - startup process 45
- compute hosts 31
 - processes running on 43
- consumer
 - purpose of 10
- consumers
 - structure 34
- CPU slots 31

D

- daemons
 - egosc 43
- dashboard
 - description of 12
- data loaders 49
- data purger 50
- data sources 49
- database
 - for reports 49
- DE
 - description of 11
- Developer Edition
 - overview 11
- directed share resource model 37

E

- EGO
 - components 30
 - description of 9
 - overview 30
- egosc 42
- executable integration with Symphony 27
- executables
 - running 27

F

- failover 46
- failover hosts 30
- fault tolerance
 - and resource management 46
 - and workload management 47
 - in applications 47
 - overview 46

H

- hierarchy of consumers 34

- hosts
 - compute 31
 - compute host processes 43
 - failover 30
 - master 30
 - master candidates 30
 - processes running on 42
 - session manager 31
 - Web server 31

I

- impersonation
 - definition of 17

K

- knowledge center
 - introduction to 14
 - overview 9

L

- lending policy 39
- lim
 - on management host 43
- loader controller 49

M

- management hosts
 - overview 30
- ManagementHosts resource group 32
- master candidates 30
- master host 30
- master lim 42
- models
 - for resource sharing 36

O

- ownership policy 39

P

- pem
 - on master host 42
- PERF 15
 - data loaders 49

- data sources 49
 - overview 49
- Platform Management Console
 - description of 9
- plc 43, 49
- PMC
 - description of 9, 12
- policies
 - borrowing 39
 - for sharing resources 38
 - lending 39
 - ownership 39
 - rank 39
 - sharing 39
- privilege
 - definition of 16
- processes
 - compute hosts 42
 - management hosts 42
 - master host 42
 - on compute hosts 43
 - on management hosts 43
 - on master candidate hosts 43
- purger 43

R

- rank policy 39
- redundancy 46
- reports 15
 - data purger 50
 - database 49
 - introduction to 15
 - loader controller 49
 - overview 9
- repository service
 - overview 26
- resource distribution plan
 - overview 31
- resource groups
 - ComputeHosts 32
 - introduction to 32
 - ManagementHosts 32
- resource management 8
- resources
 - brokered share model 37
 - CPU slots 31
 - directed share model 37

- distributing 31
- introduction to 32
- policies 38
- sharing models 36
- sharing of 38
- siloed 36
- utility model 37

role

- definition of 16

rs 42

- overview 26

S

sd 42

security

- in Symphony 17
- model 17
- user accounts 17

service

- description of 7

service controller 43

service instance

- overview 20

service instance manager

- overview 20

services

- and applications 25
- and service binaries 25
 - deployment of
 - overview 26
- starting 43

session

- overview 22

session director

- overview 20

session manager

- overview 20, 31

session manager host 31

share ratio 39

sharing policy 39

sim 43

soa applications 22

SOA middleware

- components 20

SOAM

- components of 20
- description of 9

ssm 43

start process flow 45

Symphony

- description of 6

Symphony DE

- overview 11

T

task

- overview 22

U

user roles 17

V

vemkd 42

W

Web server

- overview 31

Web server host 31

workload management 8

wsm 43