HP OpenView Service Navigator Value Pack

Service Configuration for Service Navigator Concepts Guide

Software Version: 9.0

for the HP-UX, Microsoft Windows, and Sun Solaris Operating Systems

Manufacturing Part Number: None

Document Release Date: May 2006

Software Release Date: May 2006

© Copyright 2006 Hewlett-Packard Development Company, L.P.

Legal Notices

Warranty.

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend.

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices.

©Copyright 2003-2006 Hewlett-Packard Development Company, L.P.

Trademark Notices.

Adobe®, Acrobat® are trademarks of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Intel®Celeron® Intel and Celeron are US registered trademarks of Intel Corporation.

Intel Inside Logo w/ MMX The Intel Inside Logo & Pentium are US registered trademarks and MMX is a US trademark of Intel Corporation.

Intel Itanium® Logo, Intel, Intel Inside and Itanium are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries and are used under license.

Intel® Itanium®Processor Family is a trademark of Intel Corporation in the US and other countries and is used under license.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

MMX™ is a US trademark of Intel Corporation.

MS-DOS® is a U.S. registered trademark of Microsoft Corporation.

OpenView® is a registered U.S. trademark of Hewlett-Packard Company.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

OSF, OSF/1, OSF/Motif, Motif, and Open Software Foundation are trademarks of the Open Software Foundation in the U.S. and other countries.

Pentium® is a U.S. registered trademark of Intel Corporation.

SQL*Net® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

SQL*Plus® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of the Open Group.

Windows NT® is a U.S. registered trademark of Microsoft Corporation.

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

Java™ and all Java based trademarks and logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Contents

Contents

Documentation Updates

This manual's title page contains the following identifying information:

- Version number, which indicates the software version.
- Document release date, which changes each time the document is updated.
- Software release date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition, visit the following URL:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Table 1 indicates changes made to this document since the last released edition.

Table 1 Document Changes

Chapter	Changes

Chapter 7, "Service Configuration and Service Navigator," on page 73	Added URL support for service icons and backgrounds, and graphics location directories:
	`<install_dir>\www\images\snvp\icons`
	`<install_dir>\www\images\snvp\backgrounds`
	`<install_dir>\www\images\snvp\ui`
	Added information about the following log file:
	`/var/opt/OV/log/SE_Adapter<n>.<m>.<locale>`

Support

You can visit the HP OpenView support web site at:

http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

**http://www.managementsoftware.hp.com/ **
passport-registration.html

1 Introducing Service Configuration

About Service Configuration

HP OpenView Service Configuration is an add-on product to HP OpenView Service Navigator. Service Configuration adds the following value to Service Navigator:

- **Graphical User Interface for Configuring Service Views**

  A Java-based user interface for easy configuration of objects including services, service hierarchies, status rules, actions, and many more. See “Graphical User Interface for Configuring Service Views” on page 14 for more information.

- **Mapping OVO Messages to Service Configuration Objects**

  With Service Configuration, HP OpenView Operations for UNIX messages can be easily mapped to multiple objects. See “Mapping OVO Messages to Service Configuration Objects” on page 15 for more information.

- **Service Desk Integration with Service Configuration**

  With Service Configuration installed on an existing Service Desk installation, Service Desk objects can be added as status carrying objects to Service Configuration service hierarchies. See “Service Desk Integration with Service Configuration” on page 16 for more information.

- **Service Desk Integration for the OVO Operator**

  The Service Desk integration for the OVO operator includes the following integrations:

  - Service Pages
  - Service Desk Data Forms
  - Web Console

  These applications are integrated into the Service Navigator operator console so that operators can access Service Desk data in the context of a specific service or configuration item. See “Service Desk Integration for the OVO Operator” on page 18 for more information.

Service Configuration builds on three products: OVO, Service Navigator, and Service Desk. These products may use different words to describe the same concept. See “A Word about Terminology” on page 19 for an explanation of the terms used by these products.

Graphical User Interface for Configuring Service Views

Service Configuration provides a graphical user interface (GUI) for configuring service models, including services and service resources. The GUI makes service creation fast and easy, and replaces the XML configuration files of Service Navigator which are somewhat cumbersome to use and difficult to maintain. Service Configuration not only provides all of the functionality that is already available with Service Navigator but also adds additional flexibility and functionality in the areas of mapping OVO messages to objects and working together with HP OpenView Service Desk. See “Mapping OVO Messages to Service Configuration Objects” on page 15 and “Service Desk Integration with Service Configuration” on page 16 for details.

The Service Configuration **Service Hierarchy Wizard** guides you through the steps of creating the initial definition of a service hierarchy and helps you build the infrastructure consisting of root objects, parent objects, and child objects.

Editors let you configure service hierarchies, status propagation and calculation rules, actions, and nodes.

When you are satisfied with your service hierarchy, you can easily deploy it to the corresponding OVO system where your configuration is activated and made visible to the responsible operator.

TIP

You do not have to start from nothing with Service Configuration. If you are already using Service Navigator, you can import existing Service Navigator XML configuration files into Service Configuration. See the man page *OvSnvpExport(1m)* for more information about the import tool. If you are using service discovery features offered by some HP OpenView Smart Plug-ins, the discovered output will be automatically fed into Service Configuration for you to edit. See “About Service Discovery” on page 86 for details.

See the *Getting Started with Service Configuration for Service Navigator* guide for more information about using the Service Configuration console.

Mapping OVO Messages to Service Configuration Objects

With Service Configuration, mapping OVO messages to services is fast and easy:

- **Mapping messages to multiple objects**

  When configuring a service hierarchy with Service Configuration, you simply assign strings (service names) to your objects which must be matched by OVO messages. Messages that match the name of the object or the strings set up for the object, contribute to the status of the target object or objects. One message can target multiple objects.

- **Default service names in messages**

  In OVO, message source templates format messages and thus supply values for message attributes. The content of the message attribute "service name" is therefore determined by what the OVO administrator defines in the message source template.

  If the OVO administrator has not set up any message source templates, regroup conditions, or `opcmsg` commands, OVO automatically supplies the default values for service names in messages.

See "Mapping Messages to Objects with Service Configuration" on page 67 for more information about this topic.

Service Desk Integration with Service Configuration

If you install Service Configuration on an existing Service Desk installation, you can take advantage of the integrated architecture of the two products. All objects, Service Configuration and Service Desk objects, are stored in a shared data repository and can be accessed from within Service Configuration. (You can also initially install Service Configuration standalone, with no existing Service Desk installation, and migrate to a full Service Desk environment later on.)

By providing access to a shared set of data, Service Configuration helps you combine the world of business service management (Service Desk) with the world of infrastructure service monitoring (Service Navigator). With Service Level Agreements (SLAs) mapped to real Service Navigator services and service resources, Service Desk users get access to Service Navigator troubleshooting tools such as root cause analysis and impacted services analysis.

NOTE Service Configuration (SNVP) version 9.0 is built to be used with Service Desk version 5. For more information about the SD patch level refer to the SNVP README file.

The following Service Desk object types (or “item types”) can be accessed from within Service Configuration:

- ❏ configuration item
- ❏ organization
- ❏ person
- ❏ service
- ❏ SLA (service level agreement)
- ❏ workgroup

You can add Service Desk objects of the type listed above to Service Configuration in the following ways:

- **Rule-based**

  Service hierarchy rules work like filters. They define conditions that must be met by Service Desk objects before the objects are added to a service hierarchy. For example, you could set up a filter that adds objects of the type “configuration item”, where the category equals “software”. Whenever configuration items of the category “software” are added to Service Desk, they are also added to the service hierarchy that uses the rule using the command **`Apply Service Hierarchy Rules`**.

- **Manually**

  If you only want to add a few Service Desk objects to your hierarchy, add them manually. You can also map existing Service Configuration objects to Service Desk objects.

NOTE You cannot set up Service Desk objects in Service Configuration. The objects must be set up in Service Desk before they can be added to Service Configuration.

The *Integrating Service Desk with Service Configuration for Service Navigator* guide explains how you can add Service Desk objects to Service Configuration.

Service Desk Integration for the OVO Operator

The Service Desk integration for the OVO operator includes the following integrations:

- **Service Pages integration**

  Service Pages are secure web pages that enable Service Desk customers and specialists to report problems and access data over the web. The Service Pages application is integrated with the Service Navigator operator console so that operators can access Service Desk calls, incidents, problems, changes, or workorders in the context of a specific service or configuration item. The Service Pages application is integrated in the shortcut menu of services in the Service Navigator operator console.

- **Service Desk data form integration**

  In Service Desk, detailed information about an object (or "item") is displayed in data forms. (Data forms are similar to property dialog boxes in Service Configuration.) With the Service Desk data form integration activated, operators using the Service Navigator operator console can open a data form for a Service Desk object by accessing the shortcut menu for that object.

- **Web Console integration**

  With the Web Console integration enabled, operators using the Service Navigator operator console can open a Web Console form of a selected Service Desk object by accessing the shortcut menu for that object. The Web Console Form displays the properties of a selected Service Desk object and offers a possibility to start a number of actions previously defined in Service Desk.

Refer to the Service Desk documentation for more information about Service Pages, Web Console and data forms. For details about the integrations, refer to the *Integrating Service Desk with Service Configuration* guide. The *Service Configuration for Service Navigator Installation Guide* contains detailed instructions for installing these integrations.

A Word about Terminology

Service Configuration is integrated with three products: OVO, Service Navigator, and Service Desk. These products may use different words for the same concept. Table 1-1 on page 19 gives a short overview over terminology that may give cause for confusion.

Table 1-1 Comparing Terminology

Service Configuration	Service Desk	OVO and Service Navigator
association	relation	association
attribute	field	service properties
object	item	service
object type	item type	n/a
service hierarchy	hierarchical structure	service hierarchy (Sometimes referred to as “service view”.)
service resource	configuration item	n/a

Where possible, differences in terminology are also pointed out throughout the Service Configuration documentation.

See “Glossary” on page 95 for a comprehensive list of terms.

2 Building Service Hierarchies

About Service Views

Service views let you visualize the structure of your service environment. IT organizations typically distinguish between operational and business service views. Operational service views focus on representing the structure of the IT environment. They are designed for help desk personnel, specialists, and support organizations. They are designed to show the impact of an event on a technical level. Business service views focus on showing whether a service meets the requirements that are documented in service level agreements. Business service views are designed for service delivery managers.

In Service Configuration, you create service views by building service hierarchies. You can create as many service hierarchies as you need. For example, you can create one large **service hierarchy** to display all of your services in one service view, or you can create many smaller hierarchies to display your services in different contexts. For example, you could create a service hierarchy that models a customer context and another service hierarchy that models a geographical context. The customer service hierarchy would contain all services that are used by a particular customer. The geographical service hierarchy would contain all services that are located in a particular geography. Both service hierarchies can contain the same services and service resources. But each service or service resource is considered to be an individual object with individual attributes, individual associations, individual status rules, and individual child objects. Hence the object can have a different status in different service hierarchies or contexts.

Objects make up the core data of a service hierarchy. In Service Configuration, an object can be a **service**, a **service resource**, a service **action**, or a status rule. If you use Service Configuration together with HP OpenView Service Desk, you can also include Service Desk configuration items, organizations, persons, agreements (SLAs), and workgroups. Each object can be used more than once in one service hierarchy (under different **root object**) or more than once in multiple service hierarchies:

- **Object used in one service hierarchy**

  If you use an object more than once in one service hierarchy, Service Configuration simply creates an association between the first occurrence of the object and the other occurrences. All objects share the same attributes such as name and label as well as the same associations to actions. Because all occurrences of the object also have the same status rules and the same child objects, they all have the same status in Service Navigator.

- **Object used in multiple service hierarchies**

  If you use an object more than once in different service hierarchies, for example by copying and pasting it from one hierarchy to another, Service Configuration copies the same object into each service hierarchy. Each occurrence of the object has its own, individual attributes and associations actions. Because all occurrences of the object also have individual status rules and individual child objects, they all have a different status in Service Navigator.

NOTE In the Service Navigator operator console, the operator can use the "Find in Service Views" tool to locate all occurrences of an object in different service views.

See also "Elements in a Service Hierarchy" on page 24 for more information about the elements that make up a service hierarchy.

Elements in a Service Hierarchy

Service hierarchies are organizational structures that hold objects. A **service hierarchy** is made up of the following elements:

- **Objects**

  Objects represent the core data of your service hierarchy. See “About Objects” on page 25 for details.

- **Object attributes**

  Objects are described by attributes. These attributes are presented as properties of the object in the Service Configuration console. See “About Objects” on page 25 for details.

- **Object associations**

  An object usually does not stand alone, but is associated with other objects. See See “About Object Associations” on page 28 for details.

- **Status rules**

  Status rules define how the status of an object is calculated. See See “About Status Rules” on page 29 for details.

- **Actions, nodes, and users**

  Actions, nodes, and users determine what can be done with an object, where this can be done, and who can do it. See See “About Actions, Nodes, and Users” on page 30 for details.

About Objects

In general, objects are logical or physical entities. An **object** in a Service Configuration hierarchy is usually a **service** or a **service resource**. Objects have a status that indicates their current operational state, for example the status Critical could indicate service downtime, or the status Normal could indicate service uptime. You organize objects in service hierarchies where each object assumes a certain role depending on its position in the hierarchy:

- **Root object**

  The root object is the topmost object, usually the topmost service in your service hierarchy. Root objects are always parent objects because they have child objects below them. You can have more than one **root object** in your service hierarchy, but you must have at least one.

  You can assign OVO users to root objects with the effect that the these users are responsible for the part of the service hierarchy that starts with the root object. (Root objects are not deployed to Service Navigator.)

  When you import an existing Service Navigator service configuration or when you use the service discovery features that are available with some HP OpenView Smart Plug-ins (SPIs), Service Configuration creates a root object for each existing OVO user. All objects that are assigned to a specific user are then placed into the branch of that user. Objects that are not assigned to a specific user are placed below the Lost and Found root object.

  Root objects cannot be copied. Only the branches of a service hierarchy that exist below root objects can be copied and pasted between service hierarchies. In addition, you cannot assign actions to root objects.

  Root objects cannot be mapped to Service Desk objects. This is relevant for you only if you are using Service Configuration together with Service Desk.

- **Parent object**

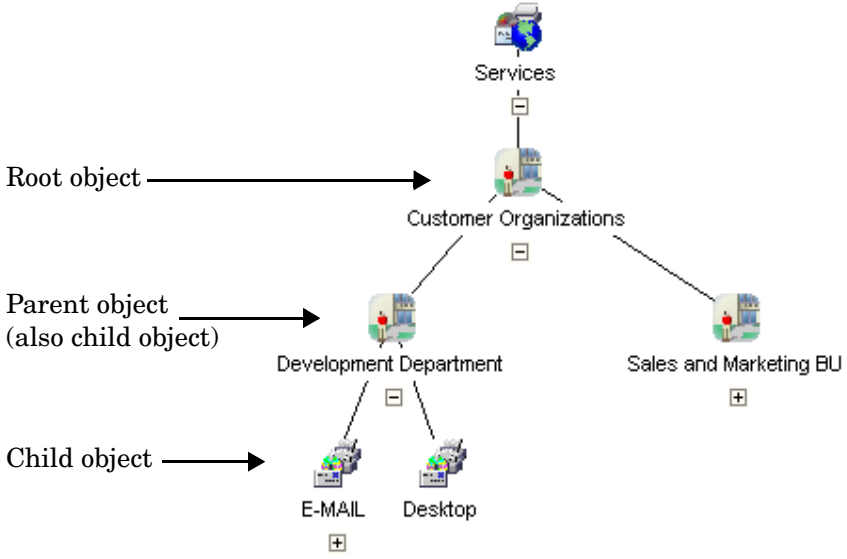
  A **parent object** has one or more child objects. If the parent object itself has a parent, this parent object is also a child object.

- **Child object**

  A **child object** has one or more parent objects. If the child object has one or more child objects, this child object is also a parent object.

Figure 2-1, “Object Roles in a Service Hierarchy,” shows a service hierarchy with objects in various roles.

Figure 2-1 Object Roles in a Service Hierarchy



See “About Object Attributes” on page 27 and “About Object Associations” on page 28 for more information about the properties and relationships of an object.

About Object Attributes

Each object is described by a set of attributes. The attributes contain the properties of the object, for example, the object name, label, and description. Other properties include, for example, the users that are assigned to the object, or the service name in message strings that map OVO messages to the object. The list of attributes varies depending on the role the object has. For example, properties of root objects include responsible users while properties of child objects include service name in message strings.

An object usually does not stand alone, but is associated with other objects. See “About Object Associations” for details.

About Object Associations

An object usually does not stand alone, but is associated with other objects. For example, a parent object can only be a parent if it is associated with a child object, and likewise a child object must have at least one parent object.

Associations include all relations an object has to other objects in Service Configuration. Since actions, hosted-on relationships, and even users are also seen as “objects” within the Service Configuration application, object relations to those “objects” also count as associations.

You can discover the associations an object has by viewing the properties of the object. Object properties also show the attributes an object has. See “About Objects” on page 25 for details.

About Status Rules

In a service hierarchy, object status is calculated from the severity of the messages targeting the object, and from the statuses of any child objects of that object. You can influence the status calculation of the object using status rules. Status rules include propagation and calculation rules. These rules are shared objects in Service Configuration which means that they can be configured globally and then be used for individual objects. See Chapter 3, “Understanding Status Calculation,” on page 33 for more information.

About Actions, Nodes, and Users

Actions are URLs or programs that, when executed on a **managed object**, either launch a web browser at a specified URL or run a **command**. You can associate actions directly with Service Configuration objects. See Chapter 4, “Understanding Actions,” on page 55 for details.

Actions are usually executed on nodes. Service Configuration distinguishes between OVO nodes and nodes that you manually add to Service Configuration. OVO nodes are managed by OVO and have an OVO agent running on them. Manually added nodes are not managed by OVO but OVO can remotely execute actions on them. Some objects are also hosted on a hosting **node**.

Service Configuration needs a list of OVO users so that it can grant them access to service views in Service Navigator. This is done by assigning users to root objects in a service hierarchy. When logged into the Service Navigator operator console, the OVO user can only operate on the relevant parts of the service hierarchy. See Chapter 6, “Service Configuration and OVO,” on page 63 for more information about importing nodes and users.

Populating Service Hierarchies

You do not need to build service hierarchies from nothing. Use the following methods to populate service hierarchies with objects:

- **Import XML configuration files from Service Navigator**

  If you already have service configuration files for Service Navigator, you can import them into Service Configuration and continue to maintain them in Service Configuration. (Service Configuration provides the command line tool `OvSnvpExport` on the OVO management server to migrate service configuration files. See the man page *OvSnvpExport (1m)* for details.

  You can import, modify, and save a Service Navigator configuration in Service Configuration, but the changes will not be visible in the Service Navigator service engine until the service hierarchy is deployed to Service Navigator.

NOTE

The data imported from Service Navigator and the data deployed from Service Configuration is not identical because the import process (the `OvSnvpExport(1m)` tool), adapts the data to fit into the Service Configuration data model. See “Service Configuration and Service Navigator” on page 73 for details.

You can save a Service Configuration service hierarchy in XML format but the syntax used is slightly different from the syntax used by Service Navigator because only a subset of the Service Navigator XML tags is used.

CAUTION

You cannot simultaneously work on a service view in the Service Navigator XML file *and* in a Service Configuration service hierarchy.

- **HP OpenView Smart Plug-in Service Discovery**

  Many HP OpenView Smart Plug-ins (SPIs) come with special discovery templates that discover objects in your environment and then automatically build a service hierarchy based on these discovered objects. You can redirect the output of these discovery processes to Service Configuration where the service hierarchy is stored in a default service view. See “About Service Discovery” on page 86 for more information.

- **Service hierarchy rules** (Service Desk integration only)

  Service hierarchy rules are part of the view builder component of Service Configuration. Service hierarchy rules work like filters—they define conditions that must be met by Service Desk objects before the objects are added to a service hierarchy. For example, you could set up a filter that adds objects of the type “configuration item”, where the category equals “software”. Whenever configuration items of the category “software” are added to Service Desk, they are also added automatically to the service hierarchy that uses the rule.

- **Manual setup**

  If you do not want to use or have access to the methods described above, you can set up a service hierarchy manually. This is more time-consuming than using automatic processes, but the Service Hierarchy Wizard is available to guide you through the required steps.

3 Understanding Status Calculation

This introduction to **status calculation** is designed to explain how the status of objects is calculated, and to show how you can influence the calculation.

If you want to create a **service model** that accurately represents the severity of the operational status of the objects you are managing, then you will need a good understanding of the calculation principles explained in this introduction.

TIP

Use the interactive status calculation tutorial to learn more about status calculation. This tutorial is only available when running the Service Configuration console on Windows. To start the tutorial, select **Help: Status Calculation Tutorial** from the menu bar.

About Object Status

The status of an object is defined by its current operational state. In Service Navigator, status colors indicate the current status, each color signalling how severe the current situation is. For example, the color red indicates a critical problem situation. Table 3-1 on page 35 shows and explains the **severity level** used by Service Navigator.

Table 3-1 Service Navigator Severity Levels

Severity Level	Color Code	Meaning
Critical	Red	Condition that affects service has occurred. Immediate corrective action is required
Major	Orange	Problem with a relatively high severity level has occurred. It is *likely* that normal use of the object will be impeded.
Minor	Yellow	Problem with a relatively low severity level has occurred. It is *unlikely* that normal use of the object will be impeded.
Warning	Cyan	Problem that affects service will or could occur. Diagnostic and corrective action is recommended.
Normal	Green	Message output is normal (that is, what was expected). For example, a process begins, a process finishes, or status information is displayed.

“About Status Rules” on page 36 explains how you can use status rules to influence the calculation of the status of an object.

About Status Rules

In a service hierarchy, the status is calculated from the severity of the messages assigned to an object, and from the statuses of any child objects of that object. You can influence the following factors within a parent-child object relationship:

- **Status propagation**

  Status propagation refers to how a child object represents its status to its parent objects. See “Status Propagation Rules Explained” on page 37 for more information about propagation rules.

- **Status calculation**

  Status calculation refers to the calculation that is performed to determine the status that is assigned to an object. This status is calculated from the severity of the messages assigned to the object itself, and from the statuses of any child objects of that object. See “Status Calculation Rules Explained” on page 41 for more information about calculation rules.

- **Weighting of child objects**

  By setting a weight you make a child object more or less important than other child objects that contribute to the parent object. See “Status Weighting Explained” on page 48 for more information about weighting.

Status Propagation Rules Explained

Status propagation refers to how a child object represents its status to its parent objects. Propagation rules are the rules that define this behavior. For example, propagation rules define whether the status of a child object is *ignored* or whether the status is *increased* to a higher level severity when calculating the status of the parent object. “Propagation Rule Options” on page 38 describes the different options you have when configuring propagation rules and “Propagation Example” on page 39 gives an example for each propagation rule option.

In Service Configuration, propagation rules are shared objects. This means that you define a rule once and then use it throughout the service hierarchies. While this saves you time, take care to use a propagation rule only in identical situations, because changes to the rule affect all objects that use the rule. A selection of the most commonly used propagation rules is available in Service Configuration to help you get started. See the *Service Configuration for Service Navigator Reference Guide* for a list of system-provided propagation rules.

Propagation Rule Options

When defining a propagation rule, you can choose between the following propagation concepts:

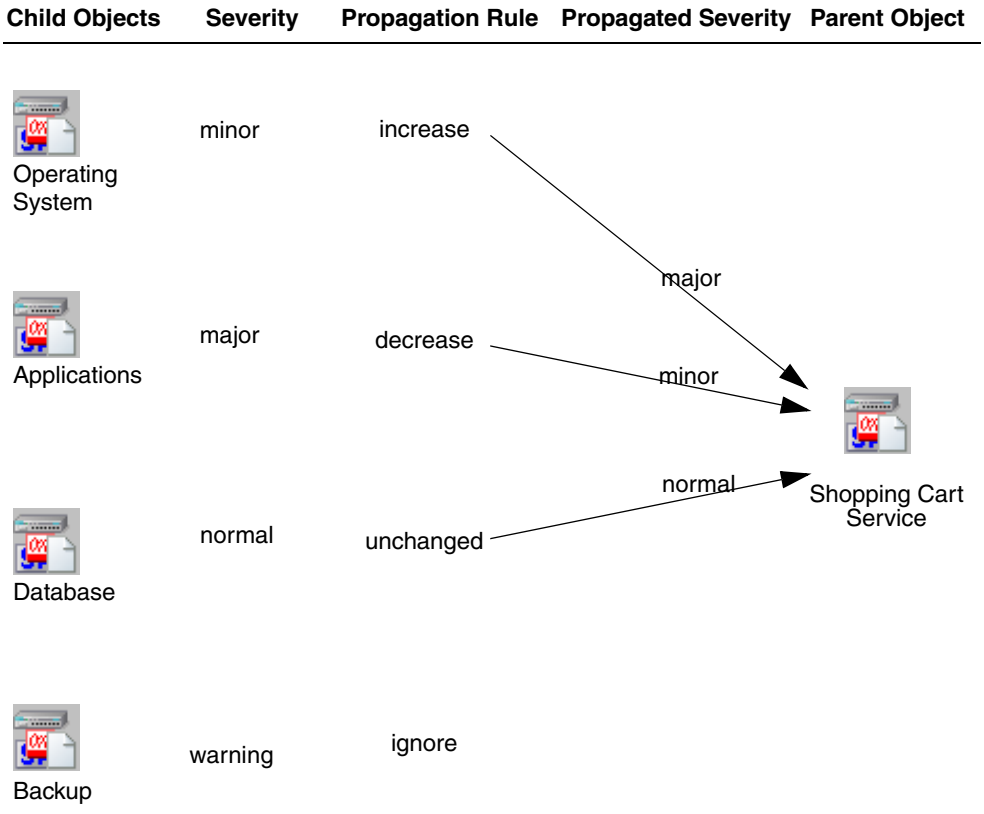
Propagation Concept	Description	Example
Unchanged	Propagate the status with no change. (This is a reasonable default value to use. If you are new to status calculation, consider starting with this value and editing it later, if necessary.)	A status of Warning equals Warning.
Ignore	The status of the child object is not considered when calculating the status of the parent object. This status propagation allows you to include a object in the service hierarchy without allowing it to influence the status calculation.	A status of Warning is ignored when calculating the status of the parent object.
Increase	The status of a child object is increased by one or more severity levels.	A status of Warning is propagated as Minor.
Decrease	The status of a child object is decreased by one or more severity levels.	A status of Warning is propagated as Normal.

See “Propagation Example” on page 39 for examples for each propagation rule option.

Propagation Example

In the example in Figure 3-1 on page 39, the child object “Applications” is less important than the child object “Operating System”. This means that child object “Applications” has a lower impact on the parent object compared to the child object “Operating System”. In this case you would want to have different propagation rules for these child objects. The status of child object “Applications” should be decreased for the purpose of status propagation, and the status of the child object “Operating System” should be increased. The child object “Database” propagates its status unchanged, while the status of the child object “Backup” does not influence the status of the parent object.

Figure 3-1 Example: Status Propagation



See “Status Calculation Rules Explained” on page 41 for information about how you can further influence the status of an object by selecting the appropriate calculation rule.

Status Calculation Rules Explained

Status calculation refers to the calculation that is performed to determine the status that is assigned to an object. This status is calculated from the severity of the messages assigned to the parent object, and from the statuses of any child objects of that parent object.

Calculation rules work on the principle that the highest level severity with a rating that reaches or crosses a threshold which you specify, is the severity of the parent object. Conceptually, Service Configuration distinguishes between different calculation rule types, which are described in more detail in “Types of Calculation Rules” on page 42. “Calculation Rule Example” on page 44 contains an example to help you better understand the concepts of status calculation.

In Service Configuration, calculation rules are shared objects. This means that you define a rule once and then use it throughout the service hierarchies. While this saves you time, take care to use a calculation rule only in identical situations, because changes to the rule affect all objects that use the rule. A selection of the most commonly used calculation rules is available in Service Configuration to help you get started. See the *Service Configuration for Service Navigator Reference Guide* for a list of system-provided calculation rules.

Types of Calculation Rules

Calculation rules let you indicate the threshold value that is used to determine the status of an object. Conceptually, Service Configuration distinguishes between the calculation algorithms described in Table 3-2.

When defining the threshold value, you can choose to perform the calculations based on a percentage-based or value-based threshold:

- **Percentage-based threshold**

  If you choose a percentage, then the threshold's values will be evaluated as percentages. The severity of the parent object changes, when the threshold is reached.

- **Count-based threshold**

  A count-based threshold interprets the thresholds as integers. The severity of the parent object changes, when the threshold is exceeded.

In most cases, a percentage threshold type is recommended because it is less rigid than a value threshold type, and therefore allows you to make changes in the number of child objects without having to change the calculation rule.

Table 3-2 Types of Calculation Algorithms

Calculation Type	Description	Example
Most Critical	The parent object adopts the status of the child object with the highest severity. The Most Critical rule models a worst-case scenario: the child object with the highest severity determines the severity of the parent object.	Each threshold is set to 0%: Critical threshold 0% Major threshold 0% Minor threshold 0% Warning threshold 0%
Single Threshold	Set the same threshold for all severities. The Single Threshold rule computes the highest average severity of all child objects. This average severity is then adopted by the parent object.	Each threshold is set to the same value: Critical threshold 50% Major threshold 50% Minor threshold 50% Warning threshold 50%

Table 3-2 Types of Calculation Algorithms (Continued)

Calculation Type	Description	Example
Multiple Thresholds	Set different thresholds for each severity. The Multiple Thresholds rule allows a few severe problems to be more important than many problems of a lower severity.	Each threshold is set to a different value: Critical threshold 20% Major threshold 40% Minor threshold 60% Warning threshold 80%

See “Calculation Rule Example” on page 44 for examples for each type of calculation rules.

Calculation Rule Example

The following example walks you through the process of calculating the status of an object. In reality, the calculation is done automatically by the Service Navigator engine, but it is important for you to understand how you can influence the calculation.

Status calculation must often account for many variables, and can become extremely complex. In order to demonstrate the concepts, this example will use only a small number of variables. The example includes the following three major steps:

Step 1: Determine the status of the child objects and messages.

Step 2: Establish the calculation matrix.

Step 3: Determine the severity by applying a calculation rule.

Go to “Step 1: Determine the status of the child objects and messages” on page 45 to begin with the example.

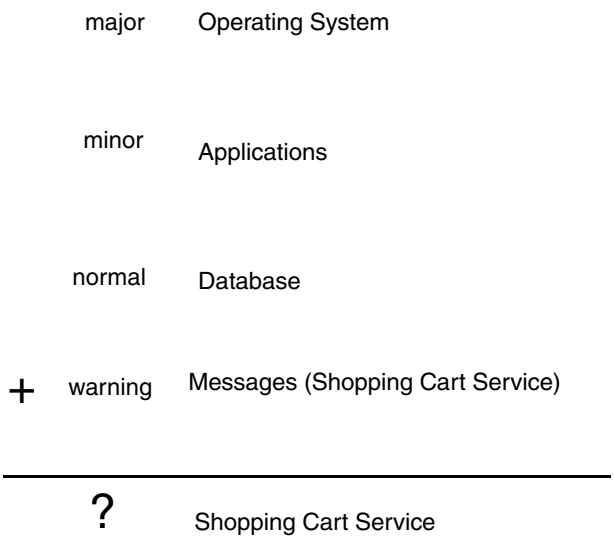
Step 1: Determine the status of the child objects and messages

Establish the propagated status of all child objects by determining the most severe active message for each child object.

Figure 3-2 shows the child objects “Operating System” (Major), “Applications” (Minor), and “Database” (Normal) with their corresponding propagated severities. The messages targeting the parent object “Shopping Cart Service” are shown with the severity Warning.

See “Status Propagation Rules Explained” on page 37 for more information about status propagation.

Figure 3-2 **Step 1: Status Calculation Example**



Go to “Step 2: Establish the calculation matrix” on page 46 to continue with the example.

Step 2: Establish the calculation matrix

Once the status of all child objects is established, calculations are performed to establish how much the severity status of each child object contributes to the severity status of the parent object.

The first part of this calculation is illustrated by the matrix shown in Table 3-3 on page 46. In the matrix, the severity status of each child object is indicated with a 1 (one). For calculation purposes, a severity status is considered to include all lower level severities, so ones (1) are added for all severities that are less than the actual severity status for that object.

The severities are rated according to the number of objects which contain the severity. The calculation is performed using the formula:

(number of objects containing the severity) / (total number of objects)

Table 3-3 Step 2: Status Calculation Example

Child Objects	**Propagated Severity**	critical	major	minor	warning	normal
Operating System	major	0	1	1	1	1
Applications	minor	0	0	1	1	1
Database	normal	0	0	0	0	1
Messages targeting the parent object	warning	0	0	0	1	1
Rating		0 (0%)	0.25 (25%)	0.50 (50%)	0.75 (75%)	1 (100%)

Go to “Step 3: Determine the severity by applying a calculation rule” on page 47 to complete the example.

Step 3: Determine the severity by applying a calculation rule

The second part of the calculation involves determining the severity of the parent object. This is done by applying one of three types of calculation rules to the calculated severity ratings. Each type of rule has a threshold for each severity rating. The highest level severity rating that exceeds its threshold becomes the severity of the "Shopping Cart Service".

Table 3-4 on page 47 compares the results of applying different types of calculation rules to the severity ratings calculated in "Step 2: Establish the calculation matrix" on page 46. Notice that the parent object adopts a different severity depending on the type of calculation rule.

Table 3-4 Step 3: Status Calculation Example

Type of Rule Applied	Resulting Severity	Details
Most Critical	major	Since Major is the highest severity found amongst the child objects, the parent object adopts the severity level Major.
Single Threshold 60%	warning	The severity that first reaches and exceeds the threshold of 60% is the severity Warning. Therefore the parent object adopts the severity Warning.
Multiple Thresholds: • Critical 20% • Major 40% • Minor 60% • Warning 80%	normal	The severity that first reaches and exceeds one of the thresholds is Normal which, being 100%, exceeds the threshold of 80%.

Status Weighting Explained

You can also give some child objects more importance than other child objects. This is done by assigning a number which indicates how much more important a particular child object should be than the other contributing child objects (for example, 2 = twice as important, 3 = three times as important). The process of assigning this number is called status weighting.

All child objects have a default weight of 1 (one). A child object's weight indicates how much the severity of that child object should contribute to the total severity. The weight proportion is calculated by dividing each child object's weight by the total number of contributing objects.

NOTE The messages assigned to an object also act as a child object and therefore have a default weight of 1.

The section “Weighting Example” on page 49 gives an example to show you how you can influence the importance of child objects.

Weighting Example

The following example shows you how the weight proportion is calculated and how the Service Navigator engine applies it when calculating the status of a parent object.

Step 1: Calculate the weight proportion.

Step 2: Establish the calculation matrix.

Step 3: Determine the severity by applying a calculation rule.

TIP In the Service Configuration console, you set the weight and calculate the weight proportion by setting a lever on a scale.

Go to “Step 1: Calculate the weight proportion” on page 50 to begin with the example.

Step 1: Calculate the weight proportion

In the following example, there are three contributing child objects and the messages targeting the parent object. If one child object is given a weight of 2 (two), the factor for that child object will be equal to 2 divided by 5, that is, 0.4. All other child objects will have a factor of 0.2. (1 divided by 5).

Table 3-5 Step 1: Status Weighting Example

Child Objects	Weight	Proportion
Operating System	1	0.2
Applications	1	0.2
Database	2	0.4
Messages	1	0.2
Total	5	1

Go to “Step 2: Establish the calculation matrix” on page 51 to continue with the example.

Step 2: Establish the calculation matrix

Once the weight proportion has been established, it is added to the calculation matrix and applied to each severity. Then the ratings are calculated. Table 3-6 on page 51 shows a calculation matrix with the proportion calculated in "Step 1: Calculate the weight proportion" on page 50.

Table 3-6 Step 2: Status Weighting Example

Child Objects	Propa-gated Severity	Weight	Propo -rtion	critical	major	minor	warning	normal
Operating System	major	1	0.2	0	0.2	0.2	0.2	0.2
Application s	minor	1	0.2	0	0	0.2	0.2	0.2
Database	normal	2	0.4	0	0	0	0	0.4
Messages targeting the parent object	warning	1	0.2	0	0	0	0.2	0.2
Rating		5	1	0 (0%)	0.2 (20%)	0.4 (40%)	0.6 (60%)	1 (100%)

Go to "Step 3: Determine the severity by applying a calculation rule" on page 52 to complete the example.

Step 3: Determine the severity by applying a calculation rule

Table 3-7 on page 52 compares the results of applying different types of calculation rules to the severity ratings calculated in Table 3-6 on page 51. Notice that the weight makes a difference for the calculation rule Single Threshold. Because the child object “Database” has more weight, it first exceeds the threshold of 60%. This is different compared to the example explained in “Calculation Rule Example” on page 44 where the messages targeting the parent object determine its severity.

Table 3-7 Step 3: Status Calculation Examples

Type of Rule Applied	Resulting Severity	Details
Most Critical	major	Since Major is the highest severity found among the child objects, the parent object adopts the severity level Major. A weight of 2 for the child object “Database” does not effect the result of applying the Most Critical rule.
Single Threshold 60%	normal	The severity that first exceeds the threshold of 60% is the severity Normal. Therefore the parent object adopt the severity warning. The child object “Database” has more weight than the messages with severity Normal.
Multiple Thresholds: • Critical 20% • Major 40% • Minor 60% • Warning 80%	normal	The severity that first exceeds one of the thresholds is Normal which, being 100%, exceeds the threshold of 80%.

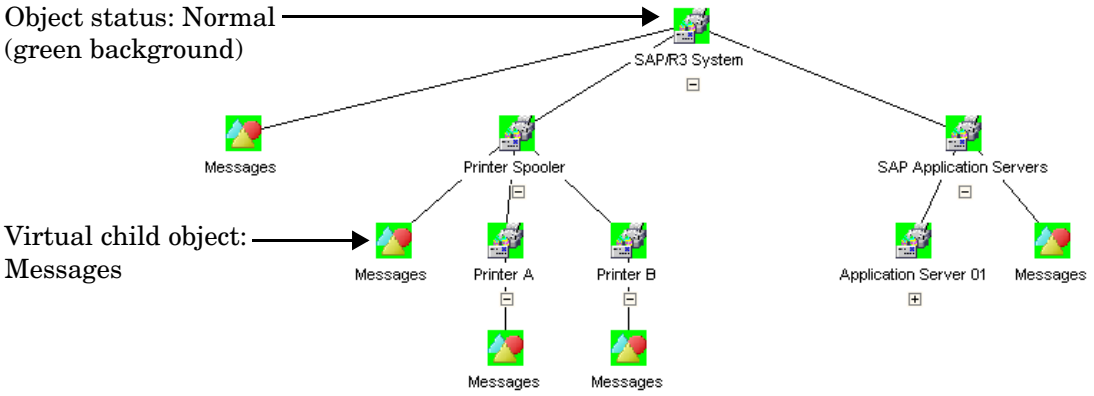
About Status Simulation

Status simulation lets you test **status calculation** in a service hierarchy. In a service hierarchy, status is calculated from the severity of the messages assigned to an object, and from the statuses of any child objects of that object. How status is calculated depends on many factors. They include the assigned status propagation and calculation rules, as well as the weights assigned to the child objects and their corresponding messages. For more information, see “Status Propagation Rules Explained” on page 37, “Status Calculation Rules Explained” on page 41 and “Status Weighting Explained” on page 48.

When simulation mode is enabled in the Service Configuration console, all objects by default assume the status Normal. This is visualized in the console by coloring the object background green and by adding a virtual child object to each object. This virtual object is called “Messages” and represents all messages that target the object. Figure 3-3 shows parts of a service hierarchy with simulation mode enabled.

“How Status Simulation Works” on page 54 explains the concepts behind simulation status in Service Configuration.

Figure 3-3 Simulation Mode Enabled



How Status Simulation Works

You simulate the status of objects by changing the **message severity level** of the messages targeting an object. These messages are represented in the virtual child object “Messages” by their service name in message attribute.

Each OVO message carries a message attribute called “service name”. To identify relevant messages, Service Configuration compares this message attribute to the following values:

- **Object name**

  A message that matches the name of the object counts in terms of status calculation.

- **Object’s service name attribute**

  These are strings you set up for the object. If the message attribute “service name” matches the object’s service name attribute, the message targets the object and its status is used to calculate the severity.

Of all matching messages, the status of the most severe active message counts.

In the example in Figure 3-4, there is a message with the severity Critical, which matches the name of the object “E-Mail”. Then there are two messages with the severities Minor and Warning. Both match a service name in message. In this example, the virtual child object “Messages” of the object “E-Mail” assumes the status Critical because Critical is the most severe message.

CAUTION

Message status information is not persistent. The message status is reset to Normal when another hierarchy is loaded, a new hierarchy is created, the hierarchy is saved under another name, the hierarchy is reloaded, and when simulation mode is disabled.

See “Mapping Messages to Objects with Service Configuration” on page 67 for more information about this topic.

4 Understanding Actions

About Actions

Actions are URLs or programs that, when executed on a **managed object**, either launch a web browser at a specified URL or run a **command**. You can associate actions directly with objects. See “Types of Commands: URL and Program” on page 58 for more information about the different types of commands that can be configured.

The benefit of associating actions with objects is that the operator can execute the action directly on the managed object by accessing a shortcut menu, without having to first select a tool in the object pane of the Service Navigator operator console. The section “Actions in the Operator Console” on page 57 contains an example to show what this looks like in the Service Navigator operator console.

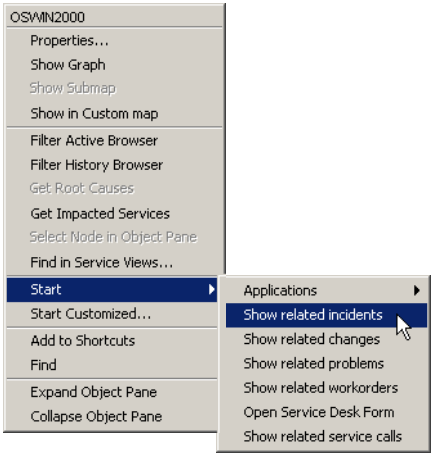
Actions are shared objects within the Service Configuration application. This means that you define an **action** only once, and then associate it with multiple managed objects. While this saves you time, take care to use actions only in identical situations, because changes to an action affect all managed objects that are associated with the action.

Actions in the Operator Console

In the Service Navigator operator console, actions appear in the shortcut menu of managed objects. If the managed object reports a problem and consequently changes its status, the operator can execute the action directly by accessing the shortcut menu, without having to first select a tool in the object pane of the Service Navigator operator console.

Figure 4-1 on page 57 shows how an operator would access an associated action. The shortcut menu for the object OSWIN2000 shows several associated actions: Show related incidents, Show related changes, Show related problems, and so on. The actions appear listed under the Start and the Start Customized menu items. The operator can now select one of the actions and execute it directly on the managed object OSWIN2000.

Figure 4-1 Accessing an Action in the Service Navigator Operator Console



See “Types of Commands: URL and Program” on page 58 for more information about action types.

Types of Commands: URL and Program

Service Configuration offers the following types of commands:

- **URL**

  Commands of the type URL start a web application. When configuring a command of the type URL, you need to supply the URL to be launched. The URL will be launched in a separate window on the system where the Service Navigator operator console is currently running.

- **Program**

  Commands of the type Program execute a command. When configuring a command of the type Program, you need to supply the command itself, the user account under which the command will be executed, and the nodes on which the command will be executed. The command can be anything, an application, a program that you have written, or a simple command that comes with the operating system.

See “Actions in the Operator Console” on page 57 for an example of how actions are used in the Service Navigator operator console.

5 Status Logging and Reporting

About Status Logging and Reporting

A critical aspect of IT management is the capability to generate reports about your service delivery performance.

If configured, Service Configuration creates a log entry in the OVO database whenever an object changes severity. Reports retrieve the logs from the OVO database and display the information in graphical or tabular form. You can design and generate your own reports or use the service reports that are bundled with HP OpenView Reporter. See “About HP OpenView Reporter Reports” on page 61 and “Designing Your Own Status Reports” on page 62 for more information.

Status logging is enabled for each object individually, either for all child objects of this object or for child objects up to a specified hierarchical level only. Status logging is disabled by default.

NOTE

See the *Service Navigator Concepts and Configuration Guide* for information about managing status logs in the OVO database. For example, you may want to download status logs from the OVO database if the amount of available disk space on the database server is no longer sufficient.

About HP OpenView Reporter Reports

Service reports cover a wide range of reports, starting from reports about the current status of an object, to reports about trends based on historical status data. The following reports are bundled with HP OpenView Reporter:

- **Service Availability**

  This report displays the duration of an object at a given operational status level. Each status is presented in percent using a pie chart. Alternatively, a bar chart can be displayed. The report is based on all status logs in the OVO database.

- **Message Trend by Service**

  This report displays the number of messages received for an object over time. It is based on all active and history messages in the OVO database that impact an object. The report displays a line graph for each object.

- **Top Active Messages by Service**

  This report displays the three objects with the highest number of active messages, broken down by severity. The report is based on all active messages in the OVO database and displays a bar chart.

See the HP OpenView Reporter documentation for more information about these reports. See “Designing Your Own Status Reports” on page 62 for more information about generating your own reports.

Designing Your Own Status Reports

Information about the status logs in the OVO database as well as instructions for designing and generating your own status reports are available in the *HP OpenView Operations Reporting and Database Schema*. HP OpenView Reporter also offers a wide range of out-of-the-box service reports. See “About HP OpenView Reporter Reports” on page 61 for more information.

6 Service Configuration and OVO

About the Relationship Between OVO and Service Configuration

The relationship between OVO and Service Configuration is twofold:

- **Importing data from OVO**

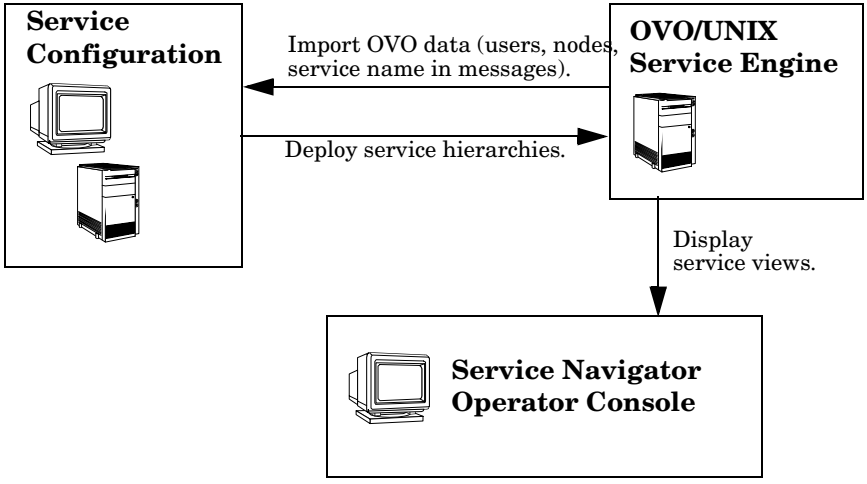
  Service Configuration requires some data from OVO to configure service hierarchies. This data is added to Service Configuration using **import** processes. See "Importing Data from OVO" on page 65.

- **Deploying service hierarchies to OVO**

  When a service hierarchy has been completely configured, it is transferred to OVO (the system where the Service Navigator service engine is running) and activated. This process is called **deployment**. Once activated, the OVO user can view the service hierarchy in the Service Navigator operator console where it is displayed as the user's service view. See also "Working with Multiple OVO Management Servers" on page 70.

Figure 6-1 on page 64 illustrates the relationship between Service Configuration, OVO, and Service Navigator.

Figure 6-1 OVO and Service Configuration Working Together



Importing Data from OVO

Service Configuration requires the following data from OVO to configure objects:

- **Nodes**

  Nodes are the target nodes for actions and are the hosting nodes for service resources. Service Configuration distinguishes between OVO nodes and nodes that you can manually add to Service Configuration. OVO nodes are managed by OVO and have an OVO agent running on them. Manually added nodes are not managed by OVO but OVO can remotely execute actions on them.

  OVO nodes are imported from the OVO database table `opc_node_names`.

- **Users**

  Only those OVO operators who are assigned to the root objects of a service hierarchy can actually access the hierarchy in the Service Navigator console. By granting operators access to selected branches of service hierarchies, you control information and data access and thereby increase operational security.

  OVO users are imported from the OVO database table `opc_user_data`.

- **Service names in messages**

  This is the message attribute “service name” that is used to map messages to objects. Service Configuration compares this **attribute** to strings that are set up for objects. If the attribute matches, the message contributes to the status of that object. (The message also contributes to the status of the object if it matches the object name.)

  Service names in messages are imported from the OVO database tables `opc_service`, `opc_act_messages`, and `opc_hist_messages`.

  See also “Mapping Messages to Objects with Service Configuration” on page 67.

OVO data is imported into Service Configuration using the Service Configuration console. When you configure an OVO management server in Service Configuration for the first time, you should also import the data from that server because it must be available before you can start configuring service hierarchies. Depending on the size of the data to be imported, the import process may take up to 30 minutes.

NOTE OVO data must be available in Service Configuration before you activate Service Configuration on the OVO management server.

Mapping Messages to Objects with Service Configuration

The messages targeting an object influence the status of the object just as other child objects do. Among all relevant messages, the message with the most severe status is the message used for status calculation. For example, if two messages target an object, one with severity Normal and another with severity Critical, the status Critical influences the severity of the object. You can also apply propagation rules and weights to messages to influence how the status of messages affects the status of an object.

Each message carries a message attribute called “service name”. To identify relevant messages, Service Configuration compares this attribute to strings (service names) you set up for the object. If the attribute and the string match, the message targets the object and its status counts. In addition, attributes that match the name of the object are also used for status calculation. One message can target multiple objects. See “About Default Service Names in Messages” on page 69 for information about how the attribute service name is set in messages.

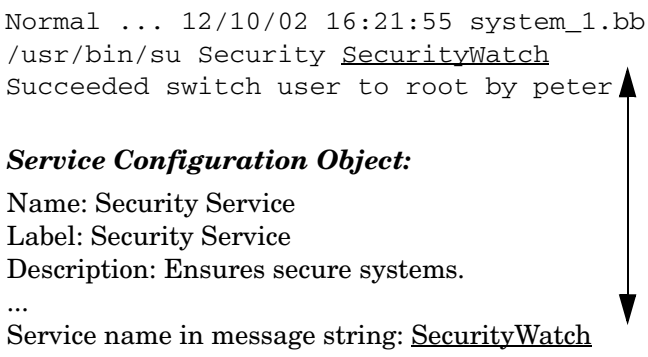
NOTE

Only service names in messages that are already imported into Service Configuration are compared in order to improve runtime performance. For configuration purposes, both active and history messages are processed by the import operation, but for status calculation purposes only active messages are used.

Figure 6-2 on page 68 shows a message with the service name attribute `SecurityWatch` and a Service Configuration object with the service name in message string `SecurityWatch`. Since both match, the message targets this Service Configuration object.

NOTE Instead of hard-coding the value of the message attribute in the service name string, you can also use wildcard characters. This gives you more flexibility for identifying relevant messages.

Figure 6-2 **Comparing Messages with Objects**



About Default Service Names in Messages

In OVO, message source templates format messages and thus supply values for message attributes. The content of the message attribute “service name” is therefore determined by what the OVO administrator defines in the message source template. This method implies that the templates and conditions have to be maintained and possibly redistributed whenever a service name changes.

NOTE Regroup conditions may override the template settings.

If the OVO administrator has not set up any message source templates, regroup conditions, or `opcmsg` commands, OVO automatically supplies the following values for the message attribute “service name”:

<*application*>:<*object*>@@<*nodename*>

Where are:

<*application*> Message attribute `application`.

<*object*> Message attribute `object`.

<*nodename*> Message attribute `nodename`.

The administrator can even set a fixed name instead of the more variable concatenation of the application, object, and node name message attributes. This is useful in situations where only some message source templates set the “service name” attribute. The administrator can choose a fixed name for all other messages and then use the Service Navigator operator console to identify these messages.

Working with Multiple OVO Management Servers

Service Configuration can work with multiple OVO management servers. This means that data from more than one OVO **management server** is added to the Service Configuration database and available for all service hierarchies you configure with Service Configuration.

One service hierarchy can use users, nodes, and service names in messages from multiple OVO management servers. The list of management servers for a service hierarchy is specified when configuring the hierarchy.

Specifying more than one management server per service hierarchy is useful when working in flexible management environments (also known as Manager-of-Manager or MoM environments) where the configuration of all participating management servers includes identical users, nodes, and messages with service name attributes. With Service Configuration, you can easily deploy the same service hierarchies to all management servers because the used users, nodes, and service names in messages are known on all management servers.

If you are working with more than one management server and your management servers do *not* share the same configuration, you must ensure that you do not mix the data of those servers in Service Configuration. This is because the **deployment** process will filter out any data that is not known on the target management server. In the worst case the deployment will fail, for example, when you deploy a service hierarchy to Server A but you only have users from Server B assigned to your root objects. In this case the assigned users are not known on the target OVO server and Service Configuration assumes that the service hierarchy is not relevant for that server.

The deployment process treats each type of OVO data differently:

- **Users and root objects**

  If the users assigned to the root objects of your service hierarchy are not known on the target management server, the service hierarchy will not be deployed at all. Your service hierarchy must contain at least one root object that is assigned to an OVO user known on the target management server. Only those parts of the service hierarchy will be deployed that start with a root object that is associated with a user who is known on the target management server.

- **Hosting nodes for resources**

  If the hosting nodes are not known on or associated with the target management server, the object references to the nodes will not be deployed. This means that all hosting information will be removed by Service Configuration.

- **Target nodes for actions**

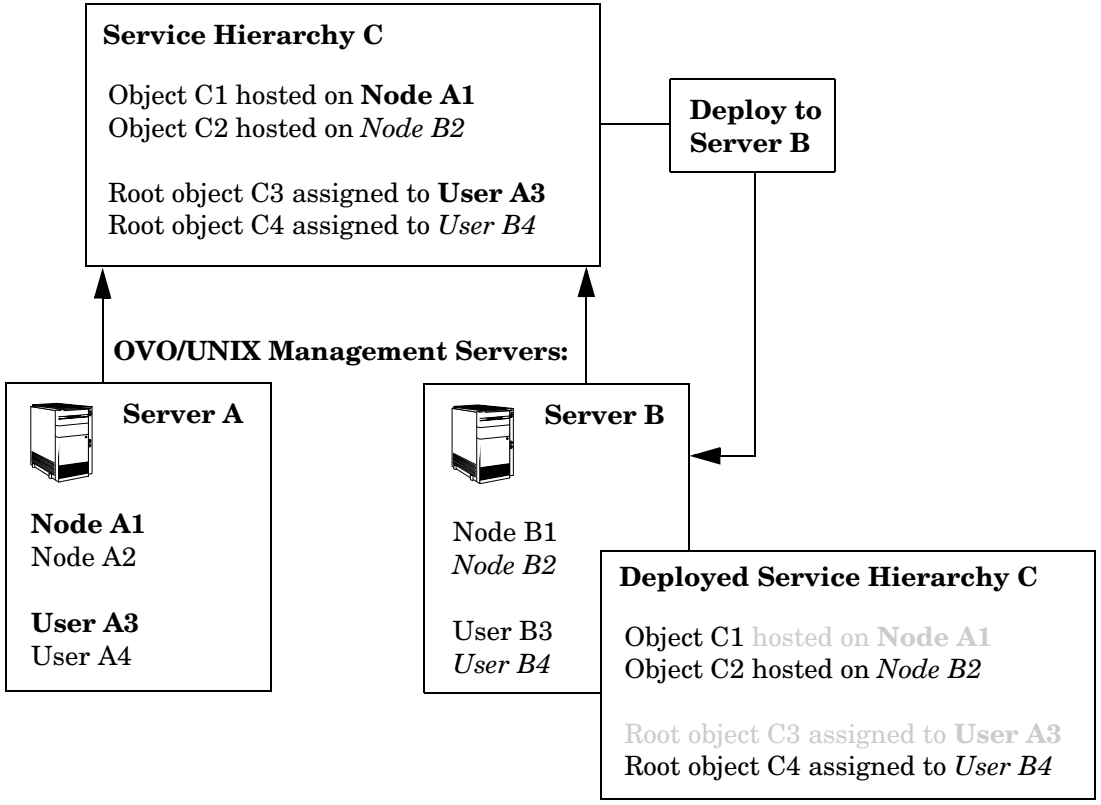
  If the target nodes are not known on or associated with the target management server, the action references to the nodes will not be deployed. This means that all target node information will be removed by Service Configuration.

- **Service names in messages**

  Before the deployment process starts, Service Configuration compares the imported service names in messages from the target management server with the strings set up for the objects in the hierarchy. All matching and all hard-coded service names in messages will be deployed. (Hard-coded service names in messages are strings that do not contain any wildcard characters.)

Figure 6-3 on page 72 shows a service hierarchy that uses data from two OVO management servers, Server A and Server B. When the service hierarchy is deployed to Server B, data from Server A is removed. What exactly is removed, depends on the type of object. Object C1 is deployed but without the hosting information because hosting node A1 is not known on Server B. The root object C3 is not deployed either because a user from Server A is assigned to it.

Figure 6-3 Deploying Service Hierarchies to Multiple Management Servers



7 Service Configuration and Service Navigator

About the Service Configuration Adapter

When you activate Service Configuration on the OVO management server, you make the Service Configuration repository the master repository for all Service Navigator configuration data. From then on, Service Configuration is your main tool for configuring Service Navigator services. After the activation, the input you supply using other Service Navigator tools, such as `opcservice`, `opcsvcterm`, or the XML Data Interface of the OVO Developer's Toolkit is diverted to Service Configuration and therefore does no longer directly affect the Service Navigator configuration. The reason for this diversion is to ensure data integrity. If multiple processes configure the service engine in parallel, they may overwrite each other's data which could result in inconsistent or even corrupt service configurations.

You can continue to use Service Navigator command line tools and interfaces to configure persistent data but your changes are not immediately visible in Service Navigator. Your changes are fed directly into the Service Configuration repository where they modify the affected service hierarchies. You must deploy the modified service hierarchies from Service Configuration to Service Navigator before your changes become visible in Service Navigator.

See Figure 7-1 on page 76 for an illustration of the Service Configuration architecture. The figure shows the following main processes:

1. **Supply input**

   Input into Service Configuration comes from the Service Navigator tools `opcservice`, `opcsvcterm`, and the XML Data Interface (available with the OVO Developer's Toolkit). Input operations that manipulate persistent data are diverted to the Service Configuration repository. Input that queries the service engine repository or that manipulates transient data continues to be processed by the service engine. See Table 7-1 on page 77 for a list of operations that are diverted to the Service Configuration repository.

2. **Redirect and migrate**

   The seadapter process diverts any input that manipulates persistent data to the Service Configuration repository. You can use the Service Configuration console to view this input and to modify it as required.

   If you have any existing service configurations in Service Navigator, you can use the command line tool `OvSnvpExport` to migrate them from the Service Navigator repository to the Service Configuration repository. See the man page *OvSnvpExport(1m)* for more information about this tool.

3. **Deploy**

   When you are satisfied with your service configuration, you can deploy it either automatically (using the console) or manually (using the command line tool `cadmsnd` on the OVO management server). (`cadmsnd` accesses the service engine directly, while `opcsvcterm` is diverted through the seadapter.) See the man page `cadmsnd`(1m) for more information about this tool. You can also use the command line tool `cadm_Deploy` to manually deploy service hierarchies. `cadm_Deploy` can be executed directly on the system where the Service Configuration console is installed.

4. **Display**

   The Service Configuration deployment process adds the deployed service configuration to the service engine repository where it is picked up the OVO Java GUI process `opcuiwww` and displayed in the Java GUI. You can now see the input you supplied in step 1 in Service Navigator.

NOTE Make sure to synchronize the Service Configuration and Service Navigator repositories through deployment as soon as possible after you have supplied input to the service engine using the standard Service Navigator tools. Otherwise any query operations do not reflect your latest changes.

Figure 7-1 The Service Configuration Architecture

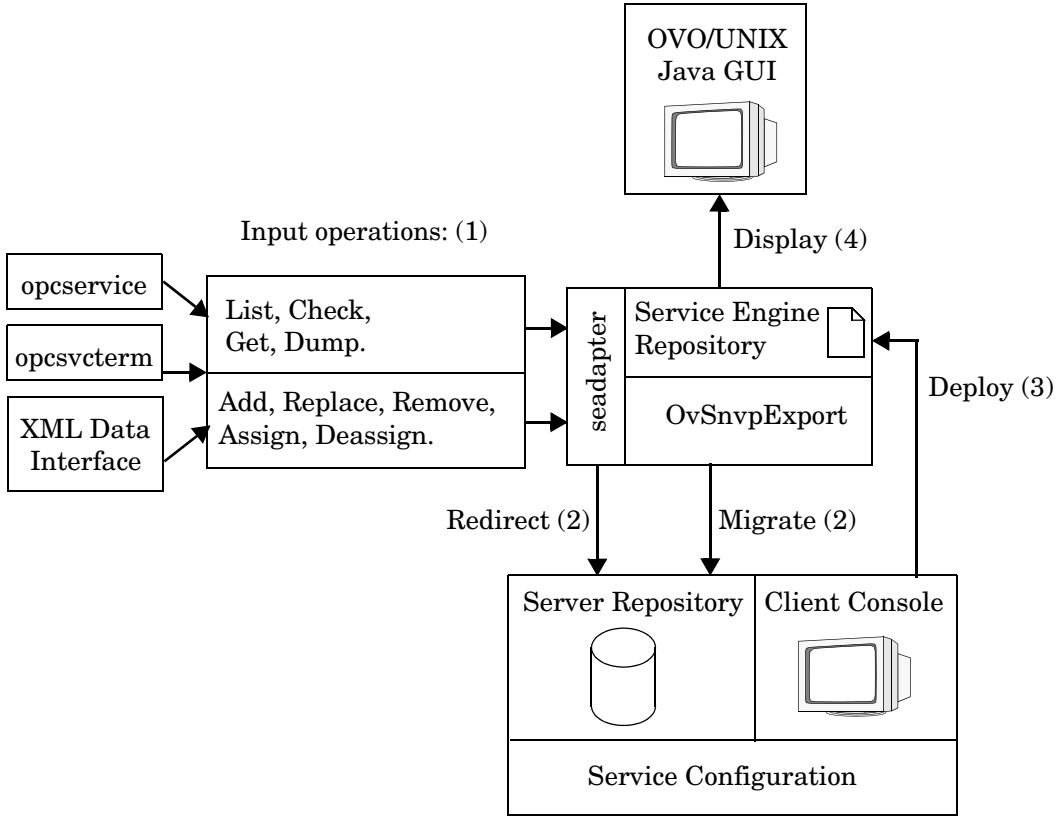


Table 7-1 Service Engine Operations that are Redirected to Service Configuration

opcservice	opcsvcterm XML Data Interface
-add	<Add>
-replace	<Replace>
-remove	<Remove>
-assign	<AssignService>
-deassign	<DeassignService>
-log_enable	<AddLoggings>
-log_disable	<RemoveLoggings>

All other operations are not affected by the Service Configuration adapter. These are operations that query the service engine (for example -list or <List>) or that change transient data (for example <SetLabel> or <SetAttributes>.

NOTE

The options -list and -check deliver unexpected results if executed directly after having modified a configuration and before having deployed it. This is because both options query the service engine repository and not the Service Configuration repository.

Similarly, you can add successive service hierarchies that depend upon each other, be aware though that the -check operation may fail to report dependencies because it queries the service engine repository and not Service Configuration.

See the section “Changes to Service Navigator Concepts” on page 78 for more information about how the Service Configuration adapter changes the behavior of Service Navigator.

Changes to Service Navigator Concepts

The Service Configuration approach to configuring services and service hierarchies differs from the Service Navigator methods. This is why the seadapter process adapts the data in the XML input stream as described in the following sections:

Use of Dependencies

Service Configuration uses only dependency relationships. When importing data from the service engine, all containment relationships are converted to dependency relationships.

This has an effect on how objects are deleted. In Service Navigator, when you delete an object that contains another object, both the containing and the contained object are deleted. With Service Configuration, an object is only deleted when it does not have any other relationships. If you delete an object that is associated with more than one parent object, only the selected object is removed. The object continues to exist in all other locations of the service hierarchy.

See also “Assignment of Users” on page 79.

Assignment of Users

In Service Configuration, you can assign users only to the top-level objects, the **root object**. When you import service hierarchies that have users assigned to lower level objects, the adapter creates a root object for each user and collects all objects that are assigned to that user below the corresponding root object. Unassigned objects are collected below the Lost and Found root object. See Figure 8-3 on page 90 for an illustration of this concept.

See also “Remove All Operations” on page 79.

Remove All Operations

In Service Navigator, the remove operation of the `opcservice` and the `opcsvcterm` tools, and of the XML Data Interface no longer globally removes objects such as “all”, “services”, “actions”, “calcrules”, “proprules”, and “operators”.

You can still remove selected objects in Service Navigator, for example, `opcservice -remove operators opc_op`. To remove large object branches or entire service hierarchies, use the Service Configuration console.

See also “Deassign All Operations” on page 80.

Deassign All Operations

In Service Navigator, the deassign operation of the `opcservice` and the `opcsvcterm` tools, and of the XML Data Interface no longer deassigns all services from an operator.

Only those root objects are deassigned that were not added by discovery processes.

See also “Remove All Operations” on page 79.

Graphics for Icons and Backgrounds

Service Configuration supports file names for graphic files for icons and backgrounds, as well as URLs. Neither relative nor absolute path names are supported.

When you migrate existing service configurations to Service Configuration, absolute and relative path names are removed, while the name of the image and URLs are preserved.

If you are using files that are not supplied with Service Navigator, you must first transfer them manually to the Service Configuration client system, and then to the OVO management server. (Graphic files are not transferred by the deployment process.)

The graphic files for icons and background graphics must be available both on the Service Configuration client and on the OVO management server in the following directories:

❑ **Graphics locations**

- *Icons directory*

  `<install_dir>\www\images\snvp\icons`

- *Backgrounds directory*

  `<install_dir>\www\images\snvp\backgrounds`

- *Images used by UI*

  `<install_dir>\www\images\snvp\ui`

Where *<install_dir>* is an SNVP installation directory (`\Program Files\HP OpenView\` by default).

NOTE

If you decide to use a graphic from the pool of graphics, you can choose a graphic for icon or background *only* from the corresponding list, that is graphic located in the corresponding directory on the Service Configuration system. You can, for example, choose a graphic for an icon only from the graphics located in the `icons` directory.

❑ **OVO management server**

`/opt/OV/www/htdocs/ito_op/images`

See also “Loggings.dtd” on page 81.

Loggings.dtd

The `operations.dtd` includes an additional DTD for loggings. The `loggings.dtd` is stored in the following directory on the OVO management server:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/dtds/loggings.dtd`

You can use this DTD to validate the results returned by the Service Navigator service engine.

See also “Check for Required Data” on page 82.

Check for Required Data

The seadapter process verifies incoming XML files to ensure that all required data is present. For example, if the XML file contains a program action without a label or a specified user, the adapter ignores the action and generates a warning message. In Service Configuration, “user” and “label” are required attributes of program actions. Note that this may cause additional errors when other objects are processed that use the ignored action.

The seadapter process logs all messages in the following log file on the OVO management server:
`/var/opt/OV/log/SE_Adapter<n>.<m>.<locale>`

Where `<n>` and `<m>` represent a logfile version, and `<locale>` represents a current locale on the system, such as `en_US` or `jp_JP`.

See also “Handling Unknown Nodes” on page 82.

Handling Unknown Nodes

The seadapter process ignores all references to nodes that are not known in Service Configuration. For example, if an imported XML file contains references to nodes that are not imported into or manually added to Service Configuration, all references to these nodes are ignored and warning messages are written to the log file `/var/opt/OV/log/SE_Adapter<n>.<m>.<locale>`. Do the following to add nodes to Service Configuration:

- **Many nodes**

  If you need to set up many nodes, set them up on the OVO management server using the OVO administrator GUI, and import them into Service Configuration. Then restart the migration process to establish the node references.

- **Few nodes**

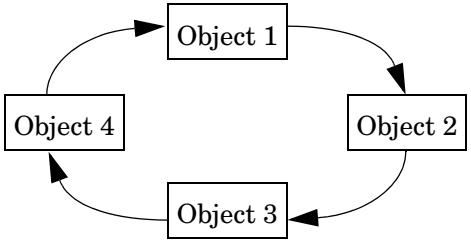
  If you only need to set up a small number of nodes, add them manually to Service Configuration using the Service Configuration Node Editor. Remember to associate them with one or more OVO management servers. Then restart the migration process to establish the node references.

See also “Handling of Cyclic Associations” on page 83.

Handling of Cyclic Associations

The seadapter process handles cyclic associations differently than a standalone Service Navigator installation. Cyclic associations are associations where an object is associated with itself either directly or indirectly. Figure 7-2 shows an example for a cyclic association: Object 1 is associated with Object 2, which is associated with Object 3. Object 3 is associated with Object 4, which is associated with the first object, Object 1.

Figure 7-2 **Cyclic Associations**



A standalone Service Navigator installation (without the Service Configuration adapter activated) ignores cyclic associations and processes the entire XML configuration file. In the example in Figure 7-2, “Cyclic Associations,” the service engine would process all associations except for the association Object 4 to Object 1. The service engine records cyclic associations as error messages (`Some associations were removed due to cyclic service structure`) in the log file `/var/opt/OV/log/OpC/mgmt_sv/opcerror`. If no other errors occur, the service configuration is processed completely and the services are added to Service Navigator.

Service Configuration handles cyclic associations differently. When the Service Configuration adapter detects a cyclic association, it stops processing and writes an error message similar to the following to standard out and to the seadapter logfile `/var/opt/OV/log/SE_Adapter<n>.<m>.<locale>`:

```
Error: A cycle of dependencies has been detected with the
following members: Object 1 - Object 2 - Object 3 - Object 4
```

The adapter operation fails completely and no objects are added to the Service Configuration repository.

See also “Labels for Objects and Actions” on page 84.

Labels for Objects and Actions

Objects and actions must have a label in Service Configuration while Service Navigator does not have this requirement. When processing XML configurations with missing labels, error messages similar to the following are generated:

```
Error: For view object the following fields are required:
label.
Error: For program action the following fields are required:
label.
```

A more detailed error message may be found in the seadapter logfile:

```
/var/opt/OV/log/SE_Adapter<n>.<m>.<locale>
```

To successfully import service configurations, either add the missing labels to your XML configurations or enable the Service Navigator compatibility mode as described in Service Configuration for Service Navigator Reference Guide.

See also “Labels With More Than 50 Characters” on page 84.

Labels With More Than 50 Characters

Service Navigator allows object labels of unlimited length while Service Configuration does not allow more than 50 characters. When migrating existing Service Navigator configurations that include objects with labels of more than 50 characters, Service Configuration uses the 50 right-most characters of the object label and discards the rest. The same applies to long labels generated by service discovery.

See also “Use of Dependencies” on page 79.

8 Working with Service Discovery

About Service Discovery

An HP OpenView **Smart Plug-in (SPI)** may provide special discovery templates that discover objects in your environment and then automatically build a service hierarchy based on these discovered objects. With Service Configuration installed and activated on the OVO management server, discovered service configurations are fed into Service Navigator, where they are diverted into Service Configuration for you to look at, modify, and activate on the OVO management server yourself.

Figure 8-1 shows the major parts of this process:

1. **Discover and add**

   SPIs discover the infrastructure, create one or more service configuration files in XML format, and then add the configuration to Service Navigator using the `opcsvcterm` or `opcservice` tools.

2. **Redirect**

   With Service Configuration installed and activated on Service Navigator, the Service Configuration adapter process (`seadapter`) redirects the input from `opcsvcterm` and `opcservice` to Service Configuration where it is placed into the repository.

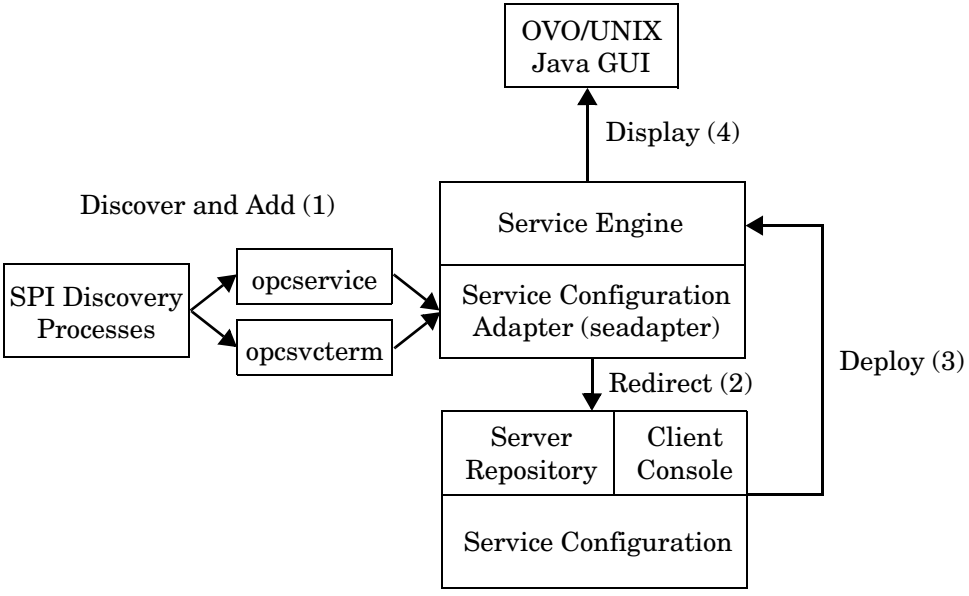
3. **Deploy**

   In Service Configuration, a special service hierarchy, the discovery hierarchy contains the discovered infrastructure. You can either deploy the discovery hierarchy immediately, without modifying it, or you can manipulate it to meet your needs. When you are satisfied with the hierarchy, you can deploy it either automatically (using the console) or manually (using the command line tools `cadmsnd` or `cadm_Deploy`).

4. **Display**

   The Service Configuration deployment process adds the deployed service hierarchy to the service engine repository where it is picked up the OVO Java GUI process `opcuiwww` and displayed in the Java GUI.

Figure 8-1 Redirecting Discovery Output



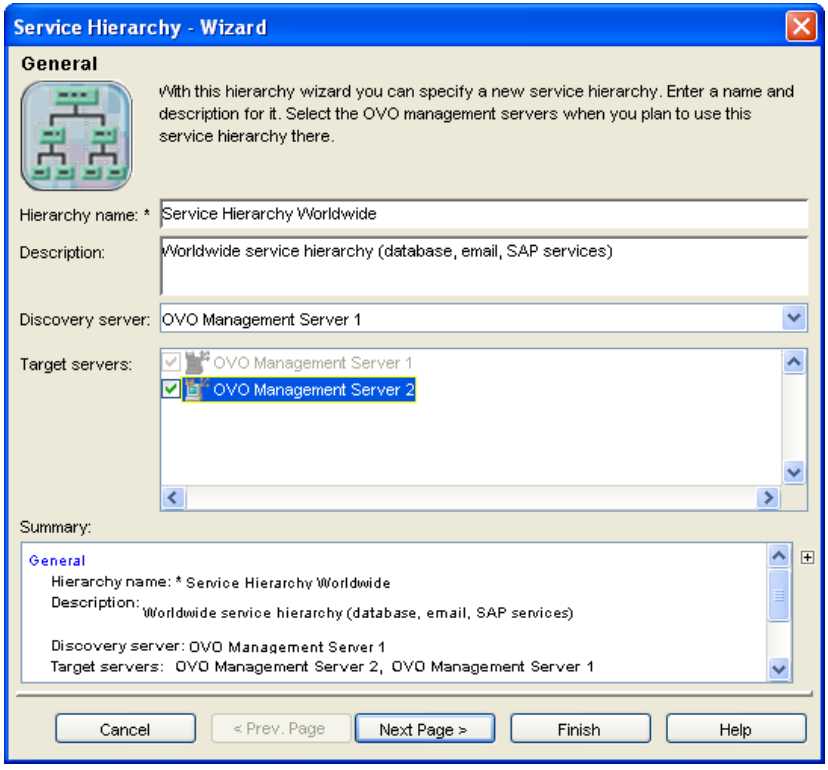
About the Default Discovery Hierarchy

Discovered service configurations are fed into a default hierarchy in Service Configuration. This is also the hierarchy where migrated Service Navigator data is stored (migrated with the tool `OvSnvpExport`).

To make a service hierarchy the default hierarchy for service discovery, choose the OVO management server where the discovery processes run as the discovery server for the hierarchy. Figure 8-2 shows the Service Hierarchy Wizard where you specify the discovery server for a service hierarchy.

For each OVO management server, you can set up only one discovery hierarchy. Note that Service Configuration also automatically selects the discovery server as target server for deployment.

Figure 8-2 **Specifying the Discovery Server**

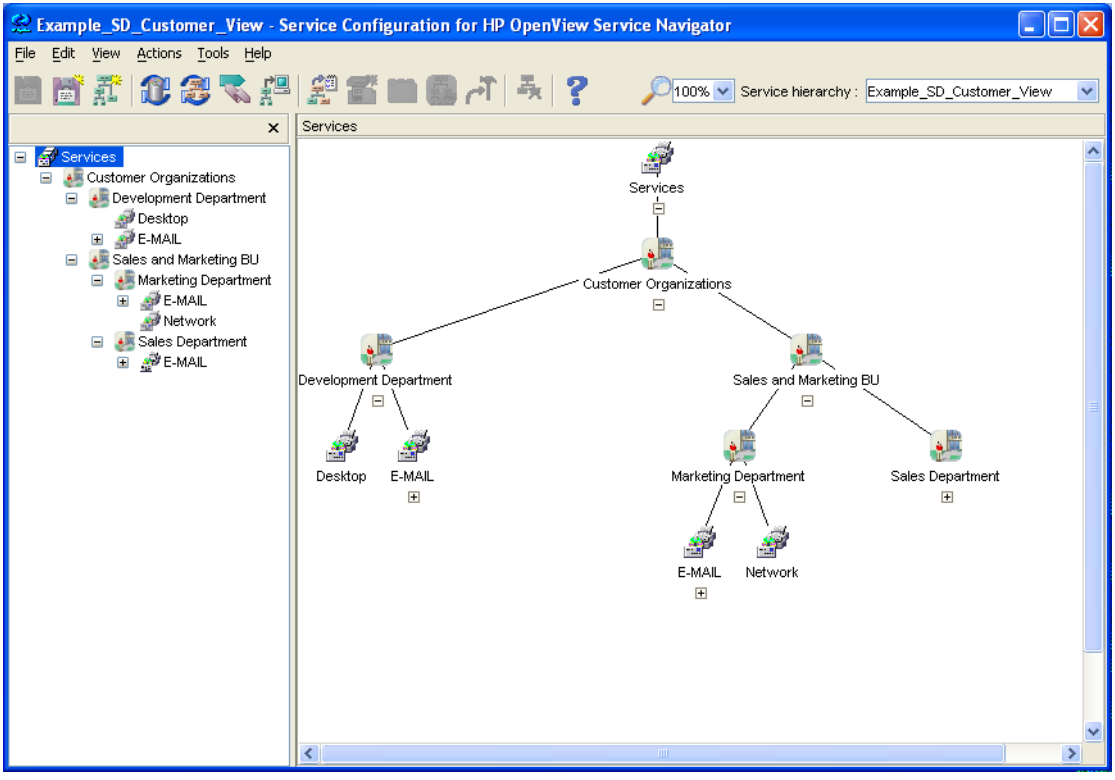


Because the data model in Service Configuration is somewhat different from the Service Navigator data model, the discovered service configuration looks different when viewed with and then deployed from Service Configuration, compared to the input generated by the SPI discovery processes. In particular, Service Configuration creates a root object for each OVO user who has objects assigned in the service configuration. All objects that are assigned to a specific user are then placed below the root object for that user. Objects that are not assigned to a specific user are placed below the Lost and Found root object. See “Service Configuration and Service Navigator” on page 73 for more information about how Service Configuration changes Service Navigator data.

Figure 8-3 shows a discovery hierarchy in Service Configuration. The hierarchy contains root objects for the OVO users opc_adm, itop, and opc_op. Objects that are not assigned to any user are placed below the Lost and Found root object.

It is up to you to decide how you want to work with the generated discovery hierarchy. You can either immediately deploy the discovered hierarchy, without modifying it, or you can manipulate it to meet your needs. See “Manipulating the Discovery Hierarchy” on page 91 and “Copying and Pasting Parts of the Discovery Hierarchy” on page 94 for more information about processing the hierarchy before deployment. See also “Service Configuration and OVO” on page 63 for more information about deploying a service hierarchy to OVO.

Figure 8-3 A Discovery Hierarchy in Service Configuration



Manipulating the Discovery Hierarchy

You can manipulate the content of the discovery hierarchy. This includes adding, excluding, or deleting objects, or modifying the attributes and associations of the objects. However, be aware that the content of the discovery hierarchy will be *overwritten* each time the discovery processes run, unless the object is write-protected by Service Configuration. Service Configuration recognizes the following protective flags:

- **Excluded from service hierarchy**

  When you exclude a discovered object from the discovery hierarchy, it does not contribute to the status of its parent object and it will not be deployed. Excluded objects are not added again by discovery processes. See “Excluding Discovered Objects” on page 92 for details.

- **Manually changed**

  When you manually change a discovered object, the object is automatically protected from further modifications through discovery processes. Otherwise it would be overwritten the next time the discovery processes run. See “Manually Changing Discovery Objects” on page 93 for details.

NOTE The discovery hierarchy does not retain knowledge about deleted discovery objects. Discovered objects that are manually deleted may be added again the next time the discovery processes run.

When you are satisfied with the discovery hierarchy, you can deploy it to Service Navigator. You have the choice between automatic and manual deployment. Automatic deployment involves using the Service Configuration main console. If you choose manual deployment, Service Configuration saves the service hierarchy to an XML configuration file which you can transfer and activate yourself.

Excluding Discovered Objects

If you choose to exclude a discovered object, the object no longer contributes status to its parent object and it will not be deployed. Discovery objects that are excluded from the discovery hierarchy will not be overwritten or removed by discovery processes because they carry a special flag which identifies them as being excluded. Figure 8-4 shows the object “Adams, Paul” with the excluded badge on the top left of the icon. Excluded objects are by default hidden in the map. The map setting Show Excluded Objects must be activated in the View menu for the excluded object to be visible.

NOTE When you manually delete a discovered object, Service Configuration will delete the object from the application. Be aware though that the next time the discovery processes discover the object again, they will add it again to Service Configuration because they have no knowledge of the fact that this object had been previously deleted.

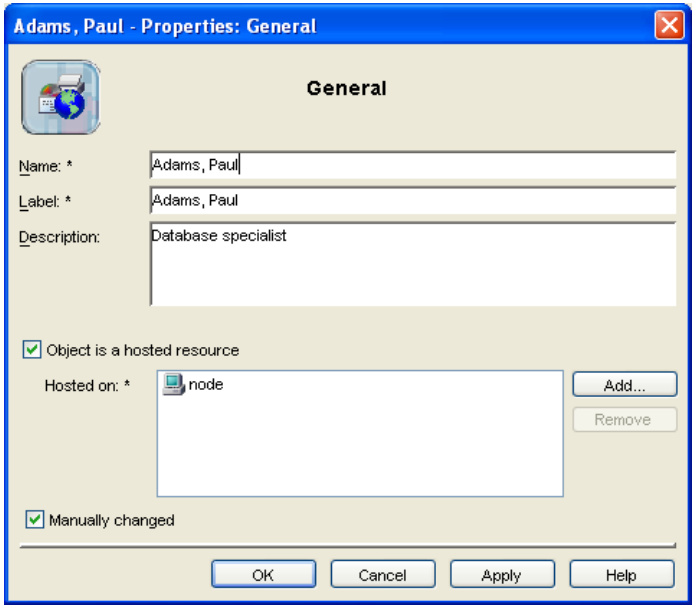
Figure 8-4 An Object Marked as Excluded

Manually Changing Discovery Objects

The first time you modify a discovered object, for example change the attributes, assign status rules and actions, and so on, Service Configuration automatically selects the check box **Manually changed** in the general properties of the object. This protects your changes from being overwritten by the discovery processes.

Figure 8-5 shows the general properties of the object “Adams, Paul”. The check box **Manually changed** is selected to indicate to the discovery processes that this object has been manually changed and should therefore not be overwritten.

Figure 8-5 The General Properties of a Child Object



TIP

If you want to place a manually added or a manually changed object under the control of service discovery, simply clear the **Manually changed** check box. The next time the discovery process detects the same object, the existing object will be overwritten with the newly found object.

Copying and Pasting Parts of the Discovery Hierarchy

If you are not satisfied with the way Service Configuration organizes the discovery hierarchy, you can also create additional service hierarchies and copy and paste branches of the discovered hierarchy into the other hierarchies.

Copied objects are no longer related to the original, discovered objects and are therefore ignored by the discovery processes. For example, if the discovered object changes in the discovery hierarchy, the copied object in the other service hierarchy is not updated.

Some of the settings of the copied objects differ from those of the original objects. These settings are compared and described in Table 8-1:

Table 8-1 Comparing Discovered and Copied Discovery Objects

Setting	Discovered Object	Copied Object
Manually changed	Initially not set. Set automatically when user changes the object manually.	Initially not set, even if set for the original object.
Association to parent object	Discovered.	Manually added.
Excluded from service hierarchy	Excluded.	Discovered objects that are excluded are not copied.

NOTE Copying and pasting large branches of an object hierarchy can be very memory-intensive. Make sure your console system meets the minimum hardware requirements. If the paste operation fails due to insufficient memory, increase the memory and try again.

Glossary

A

action An operation that can be carried out by an OpenView application. Actions are typically performed on managed objects. Actions can be manually executed by users through a menu item or toolbar buttons. Actions can also be configured to automatically occur in response to an event, message, or a change in information in the management database.

activate To make active or functional.

adapter Software that allows interoperability between two or more software products or components.

advanced customization Less common types of customization which are more flexible in their capabilities and complex in their implementation than typical customizations. As with other customizations, advanced customizations are done to meet the needs and preferences of a particular customer or user.

agent A program or process running on a remote device or computer system that responds to management requests, performs management operations, and/or sends performance and event notification. An agent can provide access to managed objects and MIB variables, interpret policy for resources and do configuration of resources.

API *See application program interface (API).*

application Packaged software that provides functionality that is designed to accomplish a set of related tasks. An application is generally more complex than a tool.

application program interface (API) An interface that enables programmatic access to an application.

ASCII American Standard Code for Information Interchange.

attribute An attribute is a characteristic or property of an object that can be described through a name value pair. An example of an attribute for a person would be Name: John Smith.

attribute name-value pair An attribute name-value pair is a combination of an attribute identifier and the value of that attribute for a specific object. An example of an attribute name-value pair for a person would be Name: John Smith.

B

badge A miniature icon, placed on or near an object's main icon, that provides information about the object's characteristics.

Boolean operator A logical operator that defines the context in which attribute values will be compared to satisfy a query or policy. The operators tell the system that:

AND	Both conditions have to be satisfied.
OR	At least one condition has to be satisfied.
NOT	No instance of this condition is allowed.

browser A module within a workspace that presents one or more views of objects and provides functionality for interacting with the objects and the views.

C

child object An entry in an object hierarchy that has a relationship to one or more parent objects within the hierarchy. A child object can at the same time be the parent of one or more child objects.

client A program or executable process that requests a service from a server

client A computer system on a network that accesses a service from another computer (server).

client console An instance of the user interface that appears on the client system while the application runs on a server.

command An instruction to a computer program that causes a specified operation to be carried out. Commands are typically typed by users on a command line.

configuration The combination of settings of software parameters and attributes that determine the way the software works, the way it is used, and/or how it appears. The combination of settings of software parameters and attributes that determine the way the software works, the way it is used, and/or how it appears.

configuration file A file that contains specifications or information that can be used for determining how a software program should look and operate.

configure To define and/or modify specified software settings to fulfill the requirements of a specified environment, application and/or usage.

console An instance of the user interface from which the user can control an application or set of applications.

customer An entity (person or organization) that receives services offered by a service provider. Typically services are provided to a customer based on a contractual relationship.

customization The process of designing, constructing and/or modifying software to meet the needs and preferences of a particular customer or user.

customize To design, construct and/or modify software to meet the needs and preferences of a particular customer or user.

D

database A repository of data that is electronically stored. Typically databases are organized so that data can be retrieved and updated.

deactivate To deliberately stop a component or object from working.

deploy To install and activate software, hardware, capabilities, and/or services so that they work in the business environment.

deployment The process of installing and activating software, hardware, capabilities and/or services so that they work in the business environment.

double-click To press and release a pointing device's button twice in rapid succession. Double-clicking is a time-dependent action. Clicking twice in the same location at slow speed (click-delay-click) is not a double-click.

dynamic parameters Parameters whose values are determined during program execution.

E

error log An output file containing error messages.

export To format and move information from the current application to a location outside the current application.

F

filter A software feature or program that functions to screen data so that only a subset of the data is presented or passed. Filters allow matching-relevant information to be extracted and acted on while non-matching-irrelevant information is held back.

find The act of seeking of specific data or objects within the management application or set management applications based on specified criteria.

G

group A collection of elements that can be treated as a unit. The set of elements in a group may be based on a rule specifying characteristics of group members or it may be arbitrary based on user selection. Assignments of associations and parameters can be made at the group level and inherited by the members of the group.

H

hierarchy Elements organized in successive levels with each lower level being subordinate to the one above.

HP OpenView A family of network and system management products, and an architecture for those products. HP OpenView includes development environments and a wide variety of management applications.

I, J, K

icon An on-screen image that represents objects that can be monitored or manipulated by the user or actions that can be executed by the user.

identifier A name that within a given scope that uniquely identifies the object with which it is associated.

import To format and move information from a location outside the current application into the current application.

install To load a product or component of a product onto a computer system or other network or system device. Installation typically involves running initial configuration scripts that are part of the installation process.

L

list A set of selectable items.

M

managed node A computer system or device in a network that is both monitored for performance, status and messages and is manipulated by means of actions in the management software.

managed object A network, system, software or service object that is both monitored for performance, status and messages and is manipulated by means of actions in the management software.

management server A server that provides management services, processes, and/or a management user interface to clients.

map A view of objects in the management environment (for example, systems, services, devices) in a format that shows one or more topological relationships between the objects.

message A structured, readable notification that is generated as a result of an event, the evaluation of one or more events relative to specified conditions, or a change in application, system, network, or service status.

message browser A graphical user interface that presents notifications that are generated as a result of an event, the evaluation of one or more events relative to specified conditions or a change in application, system, network, or service status.

message description Detailed information about an event or message.

message severity level A property of a message indicating the level of impact of the event or notification that initiated the message.

metadata Data that defines data.

monitored object A network, system, software or service object that is monitored for performance, status and messages by the management software.
Note that monitored objects cannot be manipulated by the management software.

N

node A computer system or device (for example, printer, router, bridge) in a network.

O

object A logical or physical entity that can be described by a distinct named set of attributes. Examples of objects are a customer, a service, a network, a computer, an interface, or a process.

object hierarchy Specification of parent-child relationships (in terms of grouping, not inheritance) to create a tree-like structure and the specification of views for each object in the structure. Each parent in the object hierarchy can have more than one child, and each child can have one or more parents.

object identifier (OID) A unique sequence of numbers or string of characters used for specifying the identity of an object. This identifier is obtained from either an authorized registration authority or an algorithm designed to generate universally unique values.

object rule A rule that determines what conditions must be met before an object will be classified as a child object or a parent in an object hierarchy.

object type An abstraction or categorization of objects based on the attributes of the object. Examples of object types are services, customers, hardware elements.

operator The role of a person who uses a management application to perform a specified subset of management tasks.

operator console A management console that is constrained to provide functionality that is consistent with operator level tasks.

P, Q

parameter A variable or attribute that may be given an arbitrary value for use during an execution of either a computer program or a procedure within a program.

parameter group A set of parameter values that have been categorized together on some logical basis. A parameter group can be applied to a variable as a set allowing multiple values to be assigned with one operation.

parameter type An abstraction or categorization of a parameter that determines the particular kind of data that is valid for the parameter. For example a parameter type could be IP Address which indicates that parameter values must have 4 numbers separated by decimals with the value for each number being in the range of 0 to 255.

parameter value A value that is given to a variable.

parent object An entry in an object hierarchy that has one or more child objects in the hierarchy. A parent object can at the same time be the child of another parent object.

path A sequence of directory names separated by slashes that specifies the location of any file or directory.

port A location for passing information into and out of a network device.

process A coordinated set of activities directed to achieve a business goal.

R

refresh To bring up to date. The process of refreshing synchronizes the data that is loaded in a client console with the data available on the server. Refreshing includes bringing in new data as well as removing obsolete data.

reserved word Words that have special meaning in Service Configuration and cannot be used for any other kind of identifier.

root object Topmost object or objects in an object hierarchy. A root object does not have a parent object.

S

server A program or executable process that responds to and services requests issued by clients.

server A computer system that provides a service (for example, management capabilities, file storage capabilities) to other computer systems (clients) on the network.

service A customer-based or user-oriented functionality (for example, E-mail, network bandwidth, application access) provided by a set of service resources.

service contract A contract between a service provider and a customer that is designed to create a common understanding about services, priorities, responsibilities, and price. Service contracts may or may not include Service Level Agreements (SLAs).

service hierarchy A type of object hierarchy with a service being the root object.

service level agreement (SLA) A negotiated agreement between the service provider and the customer that contains agreed upon Service Level Objectives (performance, availability, and so on) that specify the Quality of Service (QoS) to be maintained. It also contains processes and procedures to be followed when the objective is not met.

service level management The set of management functions that enables the process of measuring, reporting, and improving the quality of service being provided.

service level objective (SLO) A component within an SLA that specifies a Quality of Service metric and value that is to be met subject to specified constraints (for example, in a specified time frame).

service model A representation of the relationship between components of a service.

service offering A collection of different services that make up the products offered by a service provider. For example, a service offering can be a single service such as email or a bundled service like “web-hosting” that consists of the services: ”web page hosting”, “e-mail”, and “customer help desk”.

service provider A company or organization that provides services to customers.

service report A report that provides information about the functioning and status of one or more services.

service resource A hardware or software component or an underlying communication medium that is used to provide a service.

severity level A property of an object indicating the status of the object. Severity level is based on the impact of events or messages associated with the object.

Smart Plug-in (SPI) Prepackaged software that installs into a management console and provides management capabilities specific to a given type of business application, database, operating system, or service.

status calculation Process used to determine the status of a service or object based on the status of the source objects that contribute to the status of the service or object.

status propagation The way that the status of a status source is defined for status calculation of an object.

status propagation rule An algorithm or specification that determines how a status source will impact the overall status of an another object (for example, service, group).

status source An object or objects (for example, message, service, node, event) whose severity level or status contributes to the status of another object.

syntax The rules governing the structure and content of a language or the description of an object.

system administrator The role of a person who does configuration and maintenance on a computer system and/or the software on the system.

T

tab A page in the user interface that has a small index-card like projection. The projection typically presents the name for the page and allows navigation to the page by clicking.

target A network element, system, device, interface or software component to which management software or policies will be deployed, or on which actions will be executed.

terminate To permanently stop the execution of a computer operation.

U

UI server A server that can provide user interface components and data to multiple client consoles.

uninstall To remove previously installed software.

user A person who uses a software application, a computer or service. A user can have one or more roles. For example, a network administrator or operator role.

user configuration file A configuration file that contains specifications or information that can be used for determining how a software program should look and operate for a given user.

user ID An identifier that uniquely identifies a principal that is a person.

V

view A data presentation usually in graphical or tabular format. Views allow users to obtain information on the objects (for example, status) and may allow interaction with the objects.

view type A categorization of the presentation format for a view. View types include table, chart, graphical, tree, and explorer.

W, Z, Y, Z

workspace A working area within the OpenView Console that provides views of objects and and access to functionality from the perspective of a given task domain. The objects presented and the actions that are most readily available are scoped by the context of the task domain.

Index

Index


invent


Java™
COMPATIBLE