Oracle® Communications IP Service Activator

System Administrator's Guide

Release 7.3

E61096-03

June 2016

ORACLE®

Oracle Communications IP Service Activator System Administrator's Guide, Release 7.3

E61096-03

Contents

3 Starting and Stopping IP Service Activator

4 Setting Up Users and Menus

5 Backing Up and Restoring Data

6 Monitoring and Managing IP Service Activator

7 Monitoring and Managing the Network Processor

8 Managing the IP Service Activator Database

9 Troubleshooting IP Service Activator

10 Improving IP Service Activator Performance

Preface

This guide provides detailed instructions for monitoring and managing Oracle Communications IP Service Activator network system; including, user setup, system backup and restoration, system monitoring, database maintenance, and troubleshooting.

Audience

This document is intended for system administrators responsible for the day-to-day operation of IP Service Activator.

Before reading this guide, you should be familiar with IP Service Activator concepts. See *IP Service Activator Concepts*.

Accessing Oracle Communications Documentation

IP Service Activator for Oracle Communications documentation, and additional Oracle documentation, is available from Oracle Help Center:

http://docs.oracle.com

Related Documents

IP Service Activator System Administrator's Guide is one book in the IP Service Activator Release documentation set. For more information refer to the following documents:

- *IP Service Activator Concepts*: Describes high level concepts and the architecture of IP Service Activator.
- *IP Service Activator Installation Guide*: Provides information about the IP Service Activator directory structure and on upgrading IP Service Activator.
- *IP Service Activator User's Guide*: Provides information about setting up IP Service Activator system after installation.
- *IP Service Activator Cisco IOS Cartridge Guide*: Provides detailed technical information about features, and device configuration for Cisco IOS devices.
- Backup and Recovery procedures that come with your Oracle database: Provides details on backing up the system.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Document Revision History

The following table lists the revision history for this guide.

Version	Date	Description
E61096-03	June 2016	Documentation updates.
E61096-02	October 2015	Documentation updates.
E61096-01	July 2015	Initial release.

1

Using the Configuration GUI

This chapter explains how to use the Oracle Communications IP Service Activator Configuration GUI to manage system configuration parameters.

Overview of the Configuration GUI

The Configuration GUI simplifies the process of making configuration changes to the network by providing a central location for managing the configuration files deployed in a typical installation. The Configuration GUI reads configuration files from your IP Service Activator system and presents the data to you in a logical, readable format. If some files are not present or variables are missing, the Configuration GUI displays default values. The default values are merged with your input when you commit a new configuration.

Running the Configuration GUI

There are two modes of running the Configuration GUI client. They are as follows:

- Client (Local Host): The Configuration GUI configures only the local Windows machine. There can be only a single instance of the host.
- Remote: The Configuration GUI connects via Secure Shell (SSH) to a remote Oracle Solaris or Linux machine and manages its configuration.

In the remote mode, the server interacts with the client using Secure Shell. You should enter a user id and password that allows you to read/write configuration files. When the Configuration GUI starts the first time, it requests the remote host connection details.

> **Note:** The Configuration GUI runs automatically after the successful installation of IP Service Activator. For information about starting the Configuration GUI at a later time, see the procedures below.

To run the Configuration GUI (Solaris/Linux):

After installation, you can run the Configuration GUI from the *Service_Activator_Home*/**bin** directory.

To run the Configuration GUI (Windows):

1. Click **Start**, then select **All Programs**.

2. From the **Oracle - OracleCommunications** menu, select **Service Activator,** and then select **Configuration GUI** for remote mode or **Configuration GUI (Local Host)** for client mode.

   A Host Connection dialog box opens.

   Note: The Host Connection dialog box is available for only the remote mode.

3. On the Host Connection dialog box provide the following details for the remote machine:

   - Server Name: Unique name to identify this host
   - Host: IP address or host name of the machine
   - Remote directory: Path of the parent directory on which IP Service Activator is installed. This path is **/opt/OracleCommunications**. Any change made to this path on the Host Connection dialog box is saved in a file.
     - On Solaris/Linux, the path of the file is **WorkingData/ConfigToolTransfer/BaseHost.properties**
     - On Windows, the path of the file is **WorkingData\ConfigToolTransfer\BaseHost.properties**
   - User: User name of the remote Solaris/Linux machine
   - Authentication Type: Select the authentication type
     - Password-based: Select this to enter a password for authentication.
     - Key-based: When key-based is selected, a file with a private key must be supplied in order to authenticate. This key must match the public key on the remote SSH server.
   - Password: Password of the remote machine
   - SSH Port: Port number for SSH

   The client GUI reads these details and upon startup displays the configurations for a specific host.

4. Click **OK**.

   An information message appears indicating that the configuration is successfully loaded for the specified host.

5. Click **OK**.

   On the left-hand side panel, if it exists, a saved list of hosts on the Policy Server host appears.

6. Enter or verify the following information:

   - Client (Local Host) mode: Values for the CORBA and parameters of the Common Module Configuration parameters.
   - Remote mode: Values for CORBA, database connection information and parameters of the Common Module Configuration.

Note: IP Service Activator supports Oracle RAC failover. When an Oracle RAC database instance is no longer operational, the Policy Server automatically connects to one of the other instances.

Components Configured by the Configuration GUI

The Configuration GUI allows you to change the configuration parameters of the installed IP Service Activator components. Keep in mind that not all parameters are configured, but only the most commonly used parameters.

Table 1–1 provides a list of the configured components and a description of each component.

Table 1–1 Description of Components Configured by the Configuration GUI

Component	Description
CORBA	Naming service for communication between components. The CORBA naming service acts as an intermediary between components by keeping track of each component's naming and location information.
Database	Stores system data managed by the IP Service Activator core components (Policy Server, Network Processor, Log Collector, and so on). The database can be located on a different host than the Policy Server.
Common Module Configuration	Provides the common modules information needed by the other modules and collects database connection information.
Policy Server	Co-ordinates the access of multiple clients to a database and controls multiple distributed Network Processors.
System Log	Records system messages reported from IP Service Activator components or the managed network.
Event Handler	Collects, filters, and delivers details of faults and other events occurring anywhere in the network managed by IP Service Activator.
Transaction Monitor	Acts as a monitoring facility. Connects to the IP Service Activator Policy Server through the OSS Integration Manager (OIM) and allows upstream systems to confirm that activation transactions have been successfully applied to network elements.Transaction Monitor is turned off by default.
Log Reader	Analyzes various log files and saves log data into a database.
OSS Integration	Enables IP Service Activator to be easily integrated with OSS applications such as order entry, service assurance, fault management and billing. **Policy Services INA Integration** allows integration with the IP Naming and Addressing services of Oracle Communications Policy Services.
Network Processor Framework	Provides the framework for the specific cartridges to be used to interact with the network elements. Network Processor framework provides for installing **Service Cartridges** and configuring **Upgrade**.
Configuration Template Module	Helps to streamline the activation of services on network objects. Through the use of pre-defined or customized templates, the module extends the IP Service Activator capability to configure devices and interfaces.

Table 1–1 (Cont.) Description of Components Configured by the Configuration GUI

Component	Description
Threshold Activated Configuration Control Module	Monitors configuration changes according to your specifications, and raises a warning when thresholds are exceeded.
Web Service	Allows you to use a Web service to integrate IP Service Activator with Oracle Communications Order and Service Management (OSM)

Note: You can click in the text box of a specific port and see a list of the default port values of the configuration parameters in the Element Description panel on the Configuration GUI.

Managing Hosts with the Configuration GUI

Use the Configuration GUI to add, edit, and delete hosts and to assign different configuration groups to different hosts.

To add a host:

1. Click **Add** on the Configuration GUI to open the Host Connection dialog box.
2. Enter the connection details of the host and click **OK**.

To remove a host:

1. Select a host.
2. Click **Remove**.

Note: You cannot remove the base host.

To edit a host:

1. Select a host.
2. Click **Edit** to open the Edit Host dialog box.
3. Edit the connection details of the host and click **OK**.

Enabling and Disabling Components

The Configuration GUI displays only the components that are installed on a specific host.

To enable or disable components:

1. Select a host to display the component manager entries on the right-hand side panel of the Configuration GUI.

   The check boxes correspond to all the component entries in the **cman.cfg** file.

2. Do any one of the following:

   Select a check box to turn on the component in the **cman.cfg** file.

   Deselect a check box to turn off the component in the **cman.cfg** file.

3. Click **Validate Configuration** to validate the configurations in the GUI.

4. Click any one of the following:
   - **Commit to Host**: sends the configuration to the selected host.
   - **Save**: saves the current configuration for future reuse. Each configuration is saved on the local machine where the Configuration GUI is running.

Each configuration group contains an encrypted XML file referred to as template. The templates contain the following:

- Configuration parameters
- Default values
- Description of the parameters

Modifying Configuration Parameters

You can modify the configuration parameters.

To modify configuration parameters:

1. Select a configuration group in the host navigation tree on the left-hand side panel.
2. Edit the values on the right-hand side panel.
3. Click **Validate Configuration**.

   Note: This is optional because the validation is executed when you click Commit to Host.

4. Click **Commit to Host** or **Save**.

   For information about committing or saving, see "Enabling and Disabling Components".

Refreshing the Configuration Parameters

You can bring the latest configuration data from the host machine into the Configuration GUI.

To refresh the configuration parameters:

1. Make modifications to the configuration parameters without committing the changes.
2. Click **Refresh from Host**.

   The data available on the server is displayed on the Configuration GUI.

Managing the Configuration GUI

Use the following information to manage the Configuration GUI. The parameters available in the startup scripts are:

- **TRANSPORT_TIMEOUT** (in seconds): Connection timeout between the client and the Configuration GUI server

  On Windows: *Service_Activator_Home***Program\configGui.bat**

  On Solaris/Linux: *Service_Activator_Home***/bin/configGui.sh**

- **CONNECTION_TIMEOUT** (in seconds): Getting the database access when the system is validating a database connection

  On Windows: *Service_Activator_Home***\Program\configTool.bat**

  On Solaris/Linux: *Service_Activator_Home***/bin/configTool.sh**

The Configuration GUI manages the following configuration files:

- *Service_Activator_Home*/modules/Config/modules.properties
- *Service_Activator_Home***/Config/networkProcessor/com/metasolv/serviceactivator/networkprocessor/default.properties**
- *Service_Activator_Home***/Config/db.properties**
- *Service_Activator_Home***/Config/networkProcessor/quartz.properties**
- *Configuration_Management_Home***/Config/ConfigManagement.properties**
- *Configuration_Management_Home***/CMCollector/Collector.properties**
- *Service_Activator_Home***/Config/cman.cfg**
- *Service_Activator_Home***/odbc/network/admin/tnsnames.ora**
- *Service_Activator_Home***/Config/omniorb.cfg**
- *Service_Activator_Home***/Config/java/com/metasolv/serviceactivator/transactionmonitor/default.properties**
- *Service_Activator_Home***/Config/java/com/metasolv/serviceactivator/logreader/default.properties**
- *Service_Activator_Home***/Config/policy_server.cfg**
- *Service_Activator_Home***/Config/java/com/metasolv/serviceactivator/logreader/logCollector.properties**
- *Service_Activator_Home***/Config/networkProcessor/upgradeTool/default.properties**
- *Service_Activator_Home***/bin/integrity_checker.sh**

npSnaphot, npUpgrade and LogReader use database details mentioned in the **db.properties** file. This file is created when you run the Configuration GUI.

To create **db.properties** file:

After the installation of IP Service Activator is complete, the installer starts the Configuration GUI.

1. On the Configuration GUI, specify the database details in CORBA, Database, and Common Module Configuration parameters.

The **db.properties** file is created in the *Service_Activator_Home***/Config** folder. The **db.properties** file is used by the following applications:

- npSnapshot
- npUpgrade
- LogReader

If the **db.properties** file is not present in the *Service_Activator_Home***/Config** folder see the following procedure.

1. Go to *Service_Activator_Home*/**bin.**
2. Run configGui.sh. The Host Connection dialog box appears. Enter the database details in CORBA, Database, and Common Module Configuration parameters.
3. Click **Commit to Host**. The **db. properties** file is created in the *Service_Activator_Home*/**Config** folder.

The **cman.cfg** and **tnsnames.ora** files are not overwritten (new content is merged into the existing files), whereas the rest of the files are not overwritten upon commit.

The remote (back-end) Configuration GUI executable can be used as a Command Line Tool to copy (duplicate) a saved configuration file to another machine without using the GUI.

Syntax:

```
configTool.sh -action <set|get> -file <filename> [-password <passwd>]
```

where:

- -get option triggers a **refresh**
- -set option triggers a **commit/push**
- -filename is the same name as from the open or save dialog box in the GUI. The file must be present in the **ServiceActivator/WorkingData/ConfigToolTransfer/ directory.** For more information, see "Loading a Configuration" and "Saving a Configuration"
- -password is used to encrypt/decrypt the file. If the action is “set”, the password should match the one used when the file was generated (either in the GUI or by doing a “get” operation).

The Host /IP address is in the BaseHost.properties file. This property file is generated when Telnet session is started on GUI by setting the parameters.

```
configTool.sh -action set -file BaseHost.enc
```

Parameters can be obtained from *Service_Activator_Home***WorkingData\ConfigToolTransfer\BaseHost.properties**

```
PROP_REMOTE_DIR=/opt/OracleCommunications
PROP_SSH_PORT=22
PROP_BASHOST_DNS=10.156.66.128
```

Example:

```
./configTool.sh -action set -file BaseHost.enc
```

The user is then be prompted for a password. This command is executed on the target host.

About Configuration Files

You can use the Configuration GUI to make changes in the configuration files. You can still make the changes manually, and the latest changes can be retrieved in the GUI upon clicking **Refresh from Host**.

The configuration files are in different formats and located in different places.

Saving a Configuration

You can save the modifications made to the configuration variables. The saved configuration data can be reused.

To save a configuration file:

1. Make the appropriate modifications to the configuration variables.
2. Click **Save**. This opens the Save dialog box where you can browse for a location to save the configuration file.
3. Click **Save**. This opens the Password Setup dialog box.
4. Enter the password for encryption and confirm the password.
5. Click **OK**.

Loading a Configuration

You can load the saved configuration data.

To load a configuration file:

1. Select a host.
2. Click **Load**.

   This opens the Open File dialog box where you can browse for the saved configuration file.
3. Click **Open**.

   The Password Setup dialog box opens.
4. Enter the password. This is the same as entered for encrypting while saving the configuration file.
5. Click **OK**.

   The configuration details in the file are loaded for this host to the GUI client.
6. (Optional) Click **Validate Configuration**.
7. Click **Commit to Host.**

   The changes are sent to the selected host.

Loading a Configuration onto Another Host

You can load the configuration data of one host onto another.

To load a configuration file onto another host:

1. Select a host.
2. Change the parameters that you want.
3. Save the changes. For more information, see "Saving a Configuration".
4. Select another host.
5. Click **Load**.
6. Follow the steps in "Loading a Configuration".

Files Managed Through the Configuration GUI

You can update various configuration, configuration management, and properties files using the Configuration GUI.

The Configuration GUI tool manages the following IP Service Activator files:

- *Service_Activator_Home***/modules/Config/modules.properties**
- *Service_Activator_Home***/Config/networkProcessor/com/metasolv/serviceactivator/networkprocessor/default.properties**
- *Service_Activator_Home***/Config/db.properties**
- *Service_Activator_Home***/Config/networkProcessor/quartz.properties**
- *Service_Activator_Home***/Config/cman.cfg**
- *Service_Activator_Home***/odbc/network/admin/tnsnames.ora**
- *Service_Activator_Home***/Config/omniorb.cfg**
- *Service_Activator_Home***/Config/java/com/metasolv/serviceactivator/transactionmonitor/default.properties**
- *Service_Activator_Home***/Config/java/com/metasolv/serviceactivator/logreader/default.properties**
- *Service_Activator_Home***/Config/policy_server.cfg**
- *Service_Activator_Home***/Config/java/com/metasolv/serviceactivator/logreader/logCollector.properties**
- *Service_Activator_Home***/Config/networkProcessor/upgradeTool/default.properties**
- Also: *Service_Activator_Home***/bin/integrity_checker.sh**

Configuration Management files managed by the Configuration GUI tool:

- *Configuration_Management_Home***/Config/ConfigManagement.properties**
- *Configuration_Management_Home***/Config/ConfigManagementIpsa.properties**
- *Configuration_Management_Home***/CMCollector/Collector.properties**
- *Configuration_Management_Home***/CMCollector/CollectorIpsa.properties**

> **Note:** If you have customized any system configuration files, you must manually edit the new versions of the files after you run the Configuration GUI tool as part of the upgrade process

*.properties files:

All user changes that were made to properties files are propagated into the target file. This does not include properties that are discontinued and user comments (including property entries that are commented out).

These files are essentially rebuilt using the template file, and replacing any property values as encountered in the existing source file.

omniorb.cfg:

This file is overwritten with configuration data (which can be read from the source file and/or entered by the user through the Configuration GUI tool).

tnsnames.ora:

The **dataSourceName**, **host**, **port** and **service_name** attributes for the IP Service Activator DSN in this file are modified based on the information submitted through the Configuration GUI tool. If the DSN entry does not exist it is created.

integrity_checker.sh:

The shipped script file contains a place holder for the DSN. The configuration tool replaces this with the actual DSN as supplied by the user.

cman.cfg:

Individual entries are manipulated based on which components are installed and which are not. If there is an entry for something that is not installed or has not been selected for startup, that entry is commented out. Otherwise, an entry is added if none exists (based on the cman.cfg template). If the entry exists, it is updated based on user input.

The parameters managed in the cman.cfg file by the Configuration GUI tool are:

- orbInitialHost
- orbInitialPort
- dbSourceName
- dbUserId
- dbPasswd
- ShutdownTimeout
- NoCommandDelivery
- ORBServerPort

Command-line parameters (for example, "+debugAll") and other operational directives (e.g. startup-delay) are not managed by configuration tool. Any existing user modifications are left untouched. You should verify these values after running the tool and edit the file manually if modification are needed to match your original installation.

About the warnInOfflineMaintenanceMode Parameter

The **warnInOfflineMaintenanceMode** parameter in the network processor **default.properties** file controls whether warning messages are issued when commands are sent to a device which is set to Offline Maintenance mode in the GUI, or the Network Processor is running with -FileInterface enabled.

The default value of this parameter is **true**, indicating warnings are to be displayed.

Set the parameter value to **false** to disable the warning message for special operations such as the *bulk migration of devices to IP Service Activator management*, where numerous warning messages may be generated unnecessarily. Change it back to **true** for normal operations.

For changes to this parameter to take effect, you must restart the network processor. A device audit should always be performed before you push more configuration to a device to ensure that the device is in sync and that issues will not occur.

See "Turning Off the Offline Maintenance Mode Warning".

Encryption of Configuration GUI-managed Files

Some of the configuration files discussed in this section are encrypted and can not be manually edited. You *must* use the configuration tool for these:

- *Service_Activator_Home*/**Config/db.properties**
- *Configuration_Management_Home*/**Config/ConfigManagementIpsa.properties**
- *Configuration_Management_Home*/**CMCollector/CollectorIpsa.properties**

Configuration GUI Logging

Log files can be created for the Configuration GUI components - the server and client GUI. For more information, see "Configuration GUI Log Files".

2

Post-installation Administration Tasks

This chapter describes verification and setup tasks you may need to perform after installation is completed and you have started Oracle Communications IP Service Activator.

Verifying Network Processor Assignment and Cartridge Installation

This section provides the following procedures:

- Verifying the Network Processor is Assigned as a Proxy Agent
- Testing Cartridge Installation

For information about automatically generating options and registry files for Cisco devices, see the Cisco Cartridge Configuration Utility section in *IP Service Activator Cisco IOS Cartridge Guide*.

Verifying the Network Processor is Assigned as a Proxy Agent

The IP Service Activator GUI is adapted to display the Network Processor as a Proxy Agent under the domain. The Network Processor name in the GUI is **proxy-np-***host_name*, for example: proxy-np-srvotlab481. This procedure verifies that the Network Processor is correctly assigned as a Proxy Agent for the domain.

Pre-requisites: The Network Processor is installed and verified. The Domain is created in the GUI.

If you recently restarted the Network Processor, you may have to wait a minute for the GUI screen to register the Network Processor restart. (You can force the GUI update using the main menu command: **Tools, Find System Components**.) A blue message in the fault pane indicates that the Network Processor is running.

To assign the Network Processor as a Proxy Agent:

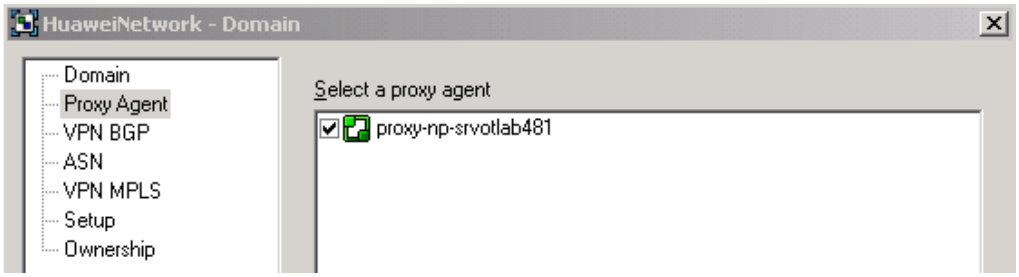
1. Log into the GUI.
2. Review the **Domain** dialog box. On the **Proxy Agent** property page, identify the **proxy-np-***host_name* object, where *host_name* is the name of the host.
3. Verify that the Network Processor is installed on that host.

   When it appears in the **Proxy Agent** dialog box, the Network Processor is confirmed as a registered component in the IP Service Activator object model.
4. Verify that the following check box is selected.

   `proxy-np-proxy_agent`

where *proxy_agent* is the name of the proxy agent.

If not, select the check box, as shown in the following image.



When the check box is selected, it confirms the Network Processor assignment as a Proxy Agent for that domain.

Testing Cartridge Installation

This section describes how to verify that cartridge installation was successful. Two approaches include:

- Viewing the Cartridge in the GUI
- Enabling Layer 2 Martini Support Through the Cisco IOS Cartridge

Viewing the Cartridge in the GUI

The cartridge appears in the GUI after the Network Processor registers the cartridge in the object model.

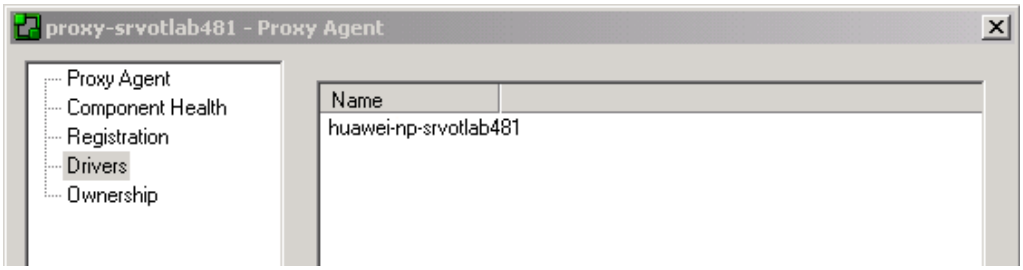
To view the cartridge in the GUI:

1. Click the **System** tab and view the **master_server** objects in the **Hierarchy** pane.

   You can see the Network Processor listed under the master_server (for example, **proxy-np-servotlab481**) and the cartridge listed in the Drivers folder (for example, **huawei-np-srvotlab481).** Discovered devices are listed last.

2. Right-click the Network Processor and select **Properties**.

   The **Proxy Agent** dialog box for the Network Processor opens. On the **Drivers** property page, note that the cartridge is listed on the right (for example, **huawei-np-srvotlab481**).



Enabling Layer 2 Martini Support Through the Cisco IOS Cartridge

Layer 2 Martini is supported through the vendor Cisco IOS cartridge and should be enabled only when required.

To enable Layer 2 Martini support:

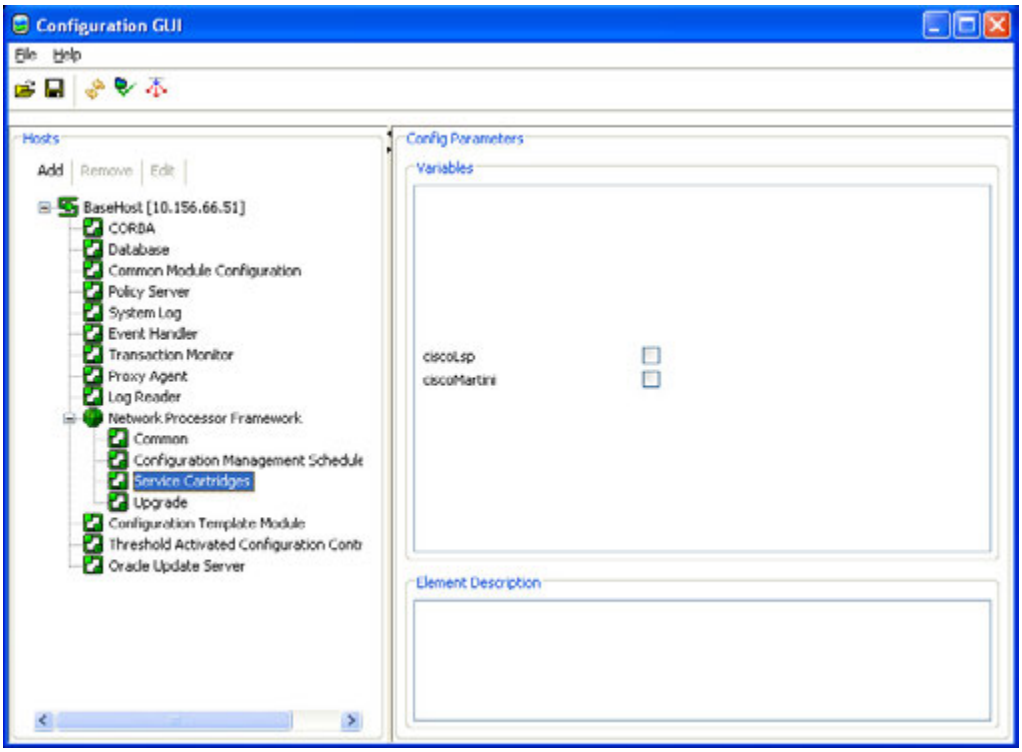
1. Log in to the Configuration GUI.

2. Under the component **Network Processor Framework**, select **Service Cartridges**.

   A list of cartridges is displayed in the **Config Parameters** panel on the right.

3. Select the **ciscoMartini** check box.

   Layer 2 Martini support is enabled.



Testing Connectivity to a Device

Cartridge connectivity to a device is verified if you can discover the device.

To discover a device:

1. On the **Global Setup** screen, view the **Domains** tab.
2. Double-click the network icon in the **Hierarchy** pane.

   The network screen appears.

3. On the **Topology** tab, select the network in the **Hierarchy** pane.
4. On the main menu, select **Discovery, and then select Discover**.

   The **Topology - Discovery** dialog box appears.

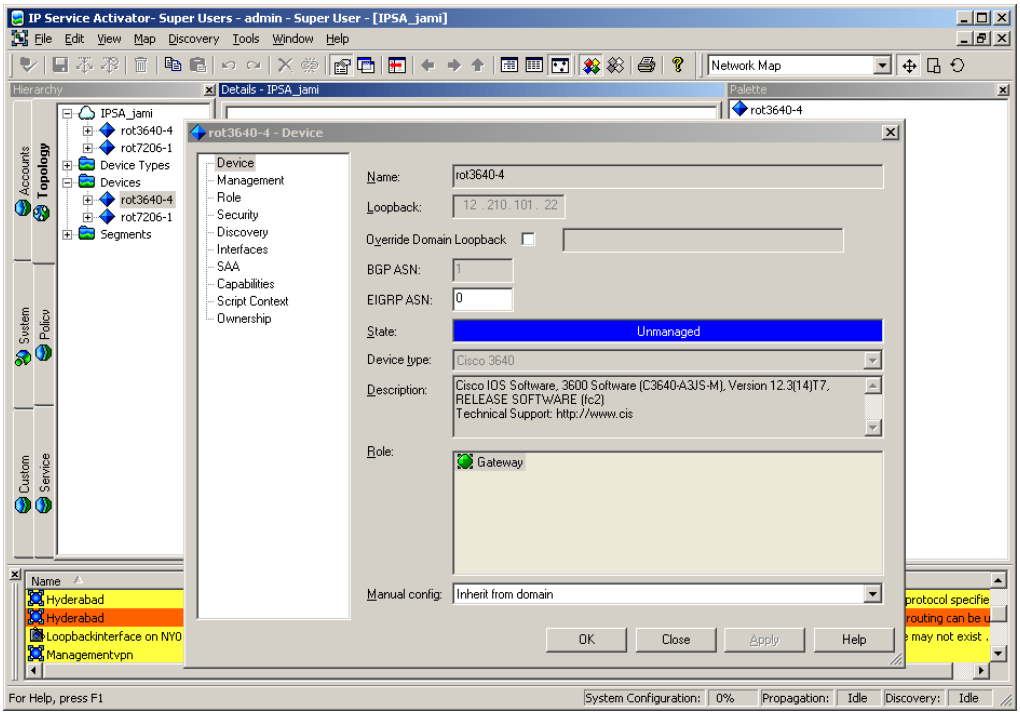
5. On the **Discovery** property page, enter the **DNS name** or **IP address** of the device that you want to discover, and then select the **Add** button.
6. On the **Security** property page, select the Access style (for example, select **TACACS+** for Huawei devices).
7. Select **OK**.
8. On the **Topology** tab, double-click the network cloud to see the discovered device.

   Each discovered device is listed under the network cloud. The device also appears in the **Details** pane, on the **Network_Name** tab.

9. Right-click the device and select **Properties**.

The **Device** dialog box appears.

The **Device** property page lists the type and a description of the discovered device.



Confirming Router Configurations

After you successfully install IP Service Activator, log into the IP Service Activator GUI and confirm the router configurations to ensure that they are set up correctly to work with the system.

All Devices

- Check that device communication protocols are enabled.
- Check that the SNMP and device security settings match the target device.

Cisco

- Check that there are some numbered access control lists (ACLs) free for use by IP Service Activator between the following ranges:
  - 100 to 199 for all Cisco devices and
  - 2000 to 2699 for Cisco devices that support the expanded range of ACLs.

  The number of free ACLs required by IP Service Activator is dependent on the number of policy rules and QoS mechanisms you want to apply to an interface.
- Check that ACLs restricting Telnet allow access from all Proxy Agents, the Policy Server and user interface hosts.
- Check that any ACLs restricting SNMP access to the router will allow access from all Proxy Agents and the Policy Server.
- Check whether there are any existing Custom Queuing, WRED, WFQ, GTS, or queue lists operating on interfaces where policy is to be applied. The IP Service

Activator system does not overwrite any pre-existing configuration with new configuration details if they conflict.

- Check whether there are route maps operating on interfaces where policy is to be applied. IP Service Activator does not update an existing route map. If you require these route maps for non-QoS purposes, you may want to use CAR-based marking on IOS 12.0 or later.

Juniper

- Check that firewall filters restricting Telnet or SSH allow access from all Proxy Agents, the Policy Server and user interface hosts.
- Check that the user specified in IP Service Activator's device security settings has access privileges that permit interface and routing configuration.
- At the system or (active) groups configuration hierarchy level, check whether there is any configuration operating on interfaces where policy is to be applied that might potentially conflict with the planned provisioning commands generated by IP Service Activator. In particular, check that there are no input or output firewall filters, VRFs, Layer 2 circuits, cross-connects and/or PHB Groups already configured on those interfaces.
- IP Service Activator detects incompatibilities in physical interface encapsulation pre-existing on the router when the cartridge is building a new configuration for CCCs and/or Layer 2 circuits. An error is displayed in the UI and the user has an option to manually correct the interface encapsulation. In order to expedite the configuration process, ensure that proper interface encapsulations are provisioned as per Juniper (Junos) documentation.

Launching the SSH Client

Secure Shell (SSH) is a UNIX-based command interface and protocol for securely accessing a remote computer. SSH commands are encrypted and both ends of the client/server connection are authenticated using a digital certificate.

Prerequisites

To run this utility, the following applications are required:

- Oracle Database is installed on a Solaris/Linux machine, and a database instance is created.
- IP Service Activator server software is installed on a Solaris/Linux machine.
- IP Service Activator GUI client software is installed on a Windows 7.0 or Windows XP Professional workstation.
- An SSH client is installed on a Windows GUI workstation.
- For client communication with a device, the SSH protocol must be enabled on the device.

For best results in the graphical user interface (GUI), it is recommended to run IP Service Activator modules and utilities at a minimum screen resolution of 1024 by 768 pixels.

The default executable for the SSH module as listed in script commands in **ExplorerScripts.xml** is **SSH.exe**. However, you must provide an actual SSH

application and update the **ExplorerScripts.xml** file accordingly, replacing **SSH.exe** with the name of the actual SSH application you are implementing.

For example, as shipped, the **ExplorerScripts.xml** file specifies **ssh.exe**:

```
<Script command="&amp;quot;C:\Program Files\Oracle Communications\IP Service
Activator/ExplorerScripts/launcher.bat&amp;quot; eeucommon.jar
com.mslv.osa.custom.executable.LaunchExec %1 %2 %3 %4 %5  start ssh.exe"
name="SSH" objects="Device"></Script>
```

You must update this line to reflect the actual SSH application that you are implementing. If using **putty**, you would change this line to read:

```
<Script command="&amp;quot;C:\Program Files\Oracle Communications\IP Service
Activator/ExplorerScripts/launcher.bat&amp;quot; eeucommon.jar
com.mslv.osa.custom.executable.LaunchExec %1 %2 %3 %4 %5  start c:\putty.exe"
name="SSH" objects="Device"></Script>
```

Procedure

To launch the SSH client:

1. From the **Topology** tab, right-click the device.
2. Select **Modules**, and then **SSH** to launch the SSH client.

About Configuration Policies

A large number of ready-to-use configuration policies are provided with IP Service Activator and can be used as-is for various purposes.

(Alternatively, you can use the IP Service Activator SDK to create and customize configuration policies as well as a service cartridge to support the implementation of the services that the configuration policy provides.)

The configuration policy files shipped with IP Service Activator are installed to the hard drive of the target machine Oracle Universal Installer.

To use the configuration policies, they must be loaded into your IP Service Activator installation through the GUI.

Only users belonging to the Read Write and Super User groups can view and use configuration policies through the GUI. See "Setting Up Users and Menus" for more information.

Refer to the chapter on configuration policies in *IP Service Activator QoS User's Guide* for more details about using configuration policies to extend the capabilities of IP Service Activator.

For detailed information about particular configuration policies, including descriptions of all fields and ranges supported on the HTML input screen, see IP Service Activator online Help.

Loading Configuration Policies

The configuration policy files included with IP Service Activator are installed to the target machine by Oracle Universal Installer during the installation of IP Service Activator.

To apply configuration policies, they must be loaded into an IP Service Activator domain in the following order:

1. **default.policy**
2. **advanced.policy**
3. **Rule_and_PHB.policy**

> **Note:** Do not load **Rule_and_PHB.policy** if you have already created standard PHB groups.

Table 2–1 lists the configuration policy files and their description:

Table 2–1 Configuration Policy File Descriptions

Filename (.policy file)	Description
advanced.policy	This file contains additional definitions for: ■ DiffServ AF, CS, BE, and EF Codepoints as specified in RFC 2474, RFC 2597, and RFC 2598 ■ MPLS Experimental values ■ Traffic Groups for Packet Marking and IP Protocols **Note:** Load **default.policy** before loading this file.
CiscoCatOSQoSPolicyTypes.policy	This file contains standard definitions for the following configuration policy: ■ Cisco CatOS Policy Rule
CiscoIOSQoSPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Cisco IOS COS Attachment ■ Cisco IOS WRR
ConfigletPolicyTypes.policy	This file contains standard definitions for the following configuration policy: ■ Configlet Policy This is a special policy type that does not include HTML. You cannot use this like other configuration policies by directly applying it on a policy target. **Note**: You must load this policy before you use Configuration Template Module (CTM).
ConfigurationManagementPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Archive Schedule ■ System Logger This is a special policy type that does not include HTML Archive Schedule.

Table 2–1 (Cont.) Configuration Policy File Descriptions

Filename (.policy file)	Description
default.policy	This file contains standard definitions for: ■ IP Precedence Codepoints ■ Standard traffic types - Gold, Silver, and Bronze ■ CoS - Gold, Silver, and Bronze ■ Well known application port numbers **Note:** ■ If you are not using the Network Processor, you may want to use **default.dscp.policy,** which is the version from earlier releases that uses DSCP values. ■ Do not load **default.policy** into a domain that has the old **default.policy** file or **default.dscp.policy**.
default.dscp.policy	This file contains standard definitions for: ■ IP Precedence Codepoints ■ Standard traffic types - Gold, Silver, and Bronze ■ CoS - Gold, Silver, Bronze ■ Well known application port numbers This variant of **default.policy** is compatible with IP Service Activator releases prior to 5.1.3. It may be more appropriate when not using the Network Processor. Packet marking policies use DSCP values, instead of IP Precedence values. **Note:** Do not load this policy file into a domain that has the new **default.policy** file.
ExtensionPackAPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ ATM PVC Vc Class ■ Save Running Config ■ Static NAT
ExtensionPackBPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ VRF Custom Naming ■ VRF Export Route Filter
ExtensionPackCPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ ATM Vc Class ■ BGP CE ■ Extended ACL
FoundryIronWareQoSPolicyTypes.policy	This file contains standard definitions for the Brocade Rate Limit configuration policy.

Table 2–1 (Cont.) Configuration Policy File Descriptions

Filename (.policy file)	Description
GeneralPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Banner ■ IP Pools ■ Key Chain ■ Prefix List ■ SNMP Community ■ SNMP Host ■ Static Route ■ User Authentication ■ User Data

Table 2–1 (Cont.) Configuration Policy File Descriptions

Filename (.policy file)	Description
InterfaceManagementPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Backup Interface Policy ■ E1 Channelized Interface ■ E1 Controller ■ E3 Channelized Interface ■ E3 Controller ■ STM1 Channelized Interface ■ STM1 Controller ■ T1 Channelized Interface ■ T1 Controller ■ T3 Channelized Interface ■ T3 Controller ■ Cisco Ethernet Port ■ Cisco Universal Interface ■ Dialer List ■ DLSW Device ■ DLSW Ethernet Interface ■ DLSW Token Ring Interface ■ HSRP ■ BRI ■ Dialer Interface ■ Loopback Interface ■ Multilink Interface ■ VirtualTemplate Interface ■ POS Interface ■ Serial Interface ■ PPP Multilink ■ SGBP ■ ATM Subinterface ■ FrameRelay Subinterface ■ VLAN Subinterface **Note:** These policies are loaded under the **Interface** sub-folder in the **Policy Types** folder.
ipsec.policy	This file contains standard definitions for: ■ IPsec proposal sets ■ IKE configurations
IPSecPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Customer IPsec ■ IPsec ■ Public IPsec

Table 2–1 (Cont.) Configuration Policy File Descriptions

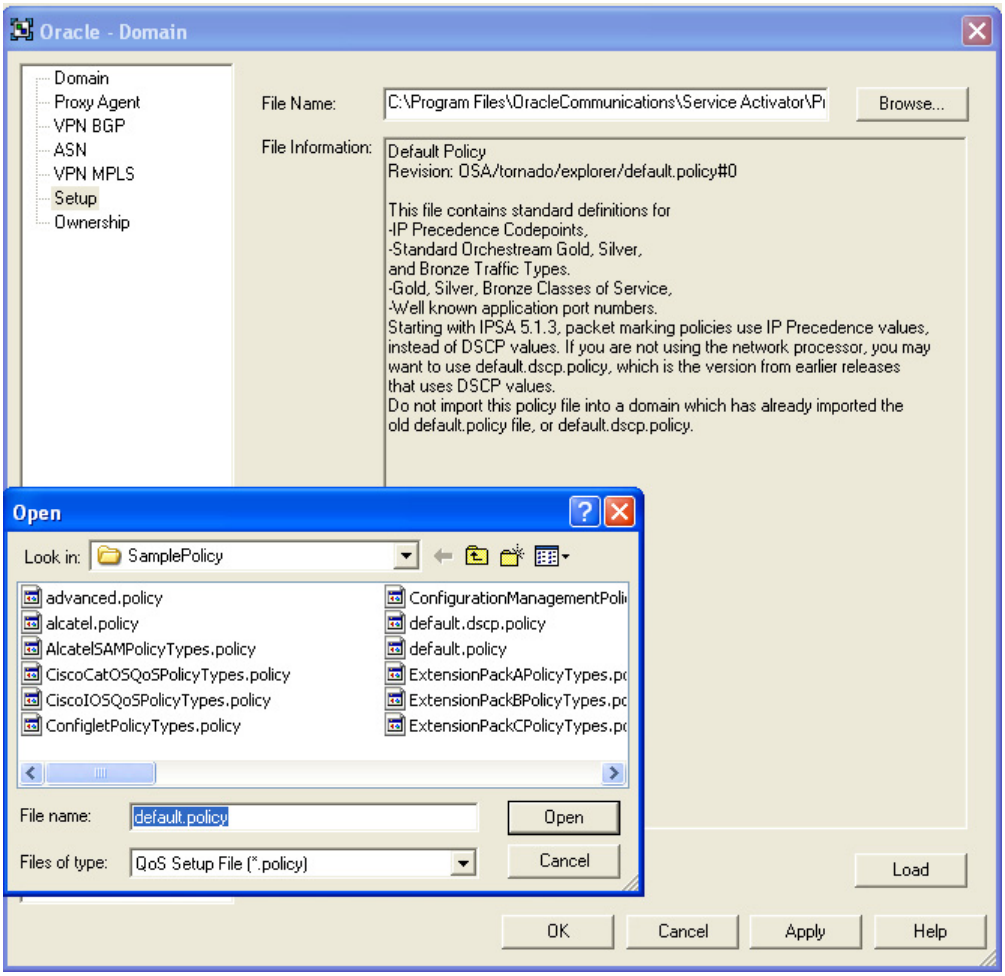
Filename (.policy file)	Description
juniper.policy	This file contains CoS definitions and an example PHB definition that uses the default IP Service Activator's IP Precedence mapping. You must load this file if you are managing QoS on Juniper M-series devices.
JuniperJUNOSQoSPolicyTypes.policy	This file contains standard definitions for the Juniper JUNOS COS Attachment configuration policy.
LSPPolicyTypes.policy	This file contains standard definitions for the LSP Tunnel configuration policy.
MulticastInterfacePolicyTypes.policy	This file contains standard definitions for the Multicast Interface configuration policy to be used with VPN and IP Multicast Policy Type services.
Role_Assignment_Rules.policy	This file contains examples of role assignment rules. **Note:** Load **default.policy** before loading this file.
RoutePolicyPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ BGP Route Policy ■ VRF Route Policy
Rule_and_PHB.policy	This file contains examples of: ■ Device and interface roles ■ PHB groups ■ Classifications ■ Access rules ■ Classification rules ■ Policing rules ■ Role Assignment rules **Note:** Load **default.policy** and **advanced.policy** before loading this file.
ServiceAssurancePolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Collector Parameters ■ Netflow Parameters ■ RTR Responder
SharedPolicyData.policy	Load this file only to add the IPv6 protocols or replace any IPv4 or IPv6 protocols that have been removed accidentally. **Note:** Your system will automatically load this file.

Table 2–1 (Cont.) Configuration Policy File Descriptions

Filename (.policy file)	Description
VLANPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ VLAN Definition ■ VLAN Interface
IPMulticastPolicyTypes.policy	This file contains standard definitions for the following configuration policies: ■ Multicast AutoRP ■ Multicast Bootstrap Router ■ Multicast Device Multicast Interface types can be installed by loading the **MulticastInterfacePolicyTypes.policy** file.
VPNMulticastPolicyTypes.policy	This file contains standard definitions for the Multicast VRF configuration policy. Multicast Interface types can be installed by loading the **MulticastInterfacePolicyTypes.policy** file.

To load configuration policies into IP Service Activator:

1. On the **Global Setup** window, select a domain from the **Hierarchy** pane.
2. Right-click on the Domain and from the pop-up menu, select **Properties**.
3. On the Domain dialog box, select the **Setup** property page.
4. Click **Browse**.



5. Select the .policy file you want to load, and click **Open**.
6. Click **Load**.

   The configuration policy loads and a confirmation dialog is displayed.
7. Click **OK** to close the confirmation dialog.
8. Click **OK** to close the Domain dialog box.
9. Commit the transaction.

After it is loaded, the configuration policy is available to all GUI instances, since it is integrated into the object model inside the domain.

You can verify which configuration policies are loaded into IP Service Activator by selecting the **Policy** tab and inspecting the contents of the **Policy Types** folder.

Adding a Configuration Policy as a New Policy Type

You can add new configuration policies from their component files (HTML and schema) when no policy files are provided.

To load a configuration policy as a new Policy Type:

1. On the **Global Setup** window, double-click a Domain from the Hierarchy pane to open it.
2. Select the **Policy** tab.

3. Expand the **Policy Types** folder.
4. Explore the hierarchy and locate the sub-folder in which you want to load the configuration policy.
5. Right-click on the target folder and select **Add Policy Type** from the pop-up menu.

   The **Policy Types** dialog is displayed.
6. In the **Name** field provide a name for the configuration policy.
7. In the **Type** field, provide the element type for the configuration policy as specified in the .xsd or as provided by the developers of the configuration policy.

   This can be determined by inspecting the .xsd file and locating the main element name near the bottom of the file. For example, the Type for the banner configuration policy is shown as:

   ```
   xs:element name="banners" type="bn:Banners"
   ```
8. Click **Import HTML**.
9. Select the HTML file for the configuration policy you are importing and click **Open**.
10. Click **Import Schema**.
11. Select the .xsd schema file for the configuration policy you are importing and click **Open**.
12. Click **OK**.
13. Commit the transaction.

    The imported configuration policy now appears in the Policy Type hierarchy.

Exporting a Policy File from a New or Customized Configuration Policy

IP Service Activator can be used to import the .html and .xsd source files into a Policy Type and then export a loadable .policy file.

Exporting the files makes it simpler to deploy configuration policies, since the policy file can be easily loaded into a domain (see "Loading Configuration Policies"). After it is loaded, the configuration policy is available to all GUI instances, since it is integrated into the object model inside the domain.

To export a policy file:

1. Follow the steps in "Adding a Configuration Policy as a New Policy Type" if the Policy Type is not already created for the configuration policy.
2. In the **Policy** tab, under the **Policy Types** hierarchy, right-click on the source configuration policy and select **Properties**.
3. Click **Export Policy**.
4. Navigate to the location you want the .policy file written to, provide a name, and click **Save**.
5. Click **OK**.

Note: The sub-folder hierarchy in place from which the source configuration policy is exported is used as the default installation location when the resultant .policy file is re-imported into the GUI.

If you want to have the configuration policy loaded to a particular sub-folder, model that location and create the Policy Type there prior to export.

Discovering Devices Using AutoDiscovery.cfg

The configuration file **AutoDiscovery.cfg**, which contains information about the devices supported and specific configuration details, is used by the IP Service Activator system when discovering devices. This file is located in the *Service_Activator_Home*/**Config** folder. It consists of a number of directives, one per line. There is one line for each vendor supported, of the following syntax:

```
Enterprise:<enterprise_number>;<enterprise_name>;<device_
driver>;<supported>;<access_type><config_level>;
```

Table 2–2 lists the parameters in the **AutoDiscovery.cfg** file for devices.

Table 2–2 Parameters in AutoDiscovery.cfg for Devices

Parameter	Description
enterprise_ number	Registered enterprise number. For example, Cisco=9.
enterprise_name	This is a text string mentioning the enterprise name.
device_driver	Name of the device driver to use support products.
supported	Whether these devices are supported (**yes** or **no**).
access_type	Default access type (one of **tacacs**, **nameduser**, **anonymous**, or **none**).
config_level	Whether interface or device level configuration (one of **device** or **interface**).

There is one line for each type of subinterface supported, of the following syntax:

```
Subinterface:<enterprise_number>;<higher_iftype>;<lower_iftype>;
```

Table 2–3 lists the parameters in **AutoDiscovery.cfg** for subinterfaces.

Table 2–3 Parameters in AutoDiscovery.cfg for Subinterfaces

Parameter	Description
enterprise_ number	Registered enterprise number. For example, Cisco=9.
higher_iftype	Interface IfType for higher layer in stack. For example, FrameRelay (32).
lower_iftype	Interface IfType for lower layer in stack. For example, AtmSubInterface (134).

A new line has been added for AutoDiscovery.cfg named **RenameSysDescr**. This can be used in the following manner:

```
RenameSysDescr:9;Engineering Special;;
```

In order for the Network Processor cartridge to allow proper parsing of the Version string **sysdescr**, and in order to infer the OS version, the Network Processor cartridge assumes that the **Version** keyword is the first one after a "**,**" character.

For further information about **AutoDiscovery.cfg**, see the Customizing discovery using AutoDiscovery.cfg section in *IP Service Activator User's Guide*.

For information about how to enable discovery of dialer interfaces with PPP encapsulation on Cisco devices, see *IP Service Activator Cisco IOS Cartridge Guide*.

Additional Setup Tasks to Be Run Prior to Initial Startup

After installing IP Service Activator and before initial startup, do the following as appropriate.

- "Permitting GUI Access to the Network Processor" (mandatory only if there is a firewall between the IP Service Activator GUI and the Network Processor): this opens the Network Processor port so the GUI can send device configuration audit results to the Network Processor.

  Note: On some Windows versions, such as Windows 7, Telnet is not installed by default. If it is not installed, you can install it by turning the Windows feature on.

- If you want to allow users to access Configuration Template Module (CTM) functionality, you can assign permissions to their IP Service Activator user groups. For more information, see "Allowing a User Group to Access Configuration Template Module" or see the information about Configuration Template Module in IP Service Activator online Help.

3

Starting and Stopping IP Service Activator

This chapter explains how to start and stop Oracle Communications IP Service Activator.

Starting IP Service Activator Components

This section describes how to start the naming service, component manager, and user interface. This can be done from the shell.

The IP Service Activator system start-up procedure requires components to be started in the following order:

1. Start the Naming Service.
2. Start the Component Managers controlling the installed Policy Server, Proxy Agent, Event Handler, and OIM components.
3. Start the User Interface.

IP Service Activator does not start if the naming service is not running, and any installed system component that does not have its component manager up and running is not available to the system. The exception is the user interface component, which does not require the component manager to be installed or running on its host machine.

During the installation, if you chose to run the naming service and component manager as services, they are started in the correct order on system reboot. You must perform this manual startup whenever you reboot the host.

Note: It is advisable to change to the **/opt/OracleCommunications/ServiceActivator/bin** directory before starting or stopping components.

Starting the Naming Service

If you chose not to run the naming service as a service that is started automatically on reboot, it must be started manually.

Note: Only the IP Service Activator administrator can start and stop the naming service.

To start the naming service from the shell:

1. Log in as the IP Service Activator administrator.

2. Type the following command:

```
$ ./ipsans start
```

This starts the naming service in the background. Logging information is recorded in **/opt/OracleCommunications/ServiceActivator/logs/ipsans.log**

To check whether the naming service is running:

1. Run the **ipsaps** script:

```
$ ./ipsaps
```

For more information about the **ipsaps** script, see "Running Troubleshooting Scripts".

Starting the Component Manager

If you chose not to run the component manager as a service that is started automatically on reboot it must be started and stopped manually.

> **Note:** Only the IP Service Activator administrator can start and stop the component manager.
>
> The naming service must be running before you can start the component manager.

To start the component manager from the shell:

1. Log in as the IP Service Activator administrator.
2. Enter the following commands:

```
/opt/OracleCommunications/ServiceActivator/bin
$ ./ipsacm start
```

This starts the component manager in the background, logging information is recorded in **/opt/OracleCommunications/ServiceActivator/logs/ipsacm.log**

To check whether the component manager is running:

1. Run the **ipsaps** script:

```
$ ./ipsaps
```

For more information about the **ipsaps** script, see "Running Troubleshooting Scripts".

To see error output:

- Type:

```
$ tail -f /opt/OracleCommunications/ServiceActivator/logs/ipsacm.log
```

To exit the log file, press **Ctrl+C**.

Starting the Windows-based User Interface

> **Note:** The naming service and component manager must be running before you can start the user interface.

To start the user interface:

- From the Start Menu select **Start**, select **Programs**, select **Oracle- Oracle Communications**, select **Service Activator**, and then select **User Interface**.

Stopping IP Service Activator Components

This section describes how to stop the user interface, component manager, and naming service. (Stopping the component manager stops other IP Service Activator components running on the same host machine.)

These activities can be performed from the shell.

Stopping the Windows-based User Interface

From the main menu, select **File**, and then select **Exit** to shut down the user interface on the Windows host.

Stopping the Component Manager

If you chose not to run the component manager as a service that is started automatically on reboot it must be started and stopped manually.

Note: Only the IP Service Activator administrator can start and stop the component manager.

The naming service must be running before you can start the component manager.

To stop the component manager from the shell:

1. Log in as the IP Service Activator administrator.
2. Enter the following command:

```
$ ./ipsacm stop
```

This stops the component manager.

To check whether the component manager is running:

1. Run the **ipsaps** script:

```
$ ./ipsaps
```

For more information about the **ipsaps** script, see "Running Troubleshooting Scripts".

To see error output:

1. Enter the following command:

```
$ tail -f /opt/OracleCommunications/ServiceActivator/logs/ipsacm.log
```

To exit the log file:

1. Press **Ctrl+C**.

Stopping the Naming Service

If you chose not to run the naming service as a service that is started automatically on reboot it must be started and stopped manually.

Note: Only the IP Service Activator administrator can start and stop the naming service.

To stop the naming service from the shell:

1. Log in as the IP Service Activator administrator.
2. Type the following command:

```
$ ./ipsans stop
```

This stops the naming service.

To check whether the naming service is running:

1. Run the **ipsaps** script:

```
$ ./ipsaps
```

For more information about the **ipsaps** script, see "Running Troubleshooting Scripts".

To see error output:

- Type:

```
$ tail -f /opt/OracleCommunications/ServiceActivator/logs/ipsans.log
```

  To exit the log file, press **Ctrl+C**.

4

Setting Up Users and Menus

This chapter explains tasks you need to do immediately after installation to set up user groups and users. It also describes other menu setup tasks and tools.

About Users and Security

Oracle Communications IP Service Activator allows you to control who has access to the system, and at what level. Access to the IP Service Activator user interface is controlled using user names and passwords. Every user who wants to gain access must be set up as a user with a user name and password. The ability to create new users and assign passwords is restricted to users with Super User access, the highest access level in IP Service Activator.

The actions a user can perform in IP Service Activator are dictated by the group to which he or she belongs. Every user is a member of a user group and the access rights specified for the group apply to the members of the group. The following group access levels are available:

- **Read Only**: group members can view objects but not modify them
- **Read Write**: group members can view or modify objects
- **Super User**: group members can view and modify all objects

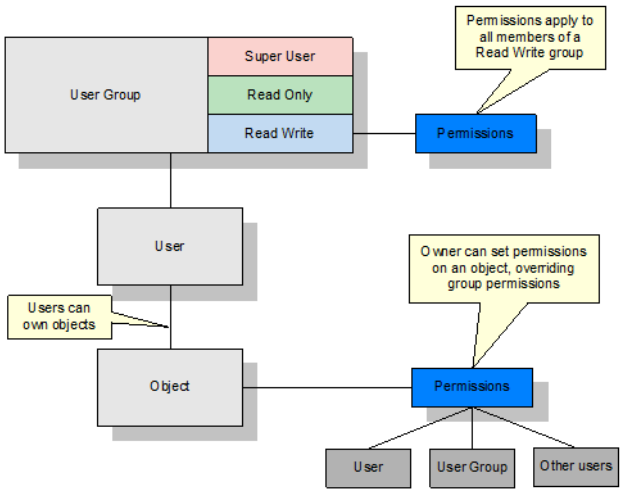
For Read Write groups, you can specify exactly which elements of IP Service Activator its members can view and modify using permissions. This enables you to limit access according to the role a user performs, or the customer accounts he or she is managing. When you first install IP Service Activator, you might need to create a number of Read Write groups to apply the necessary access levels to all users.

> **Note:** A user's access level affects the appearance of the user interface. For example, a Read Write user with access barred to policy cannot see the **Policy** tab or access menu items that relate to policy.

IP Service Activator also offers a concept of object ownership that allows a user to own specific objects and set permissions that apply to themselves, the group to which they belong, and all other users.

Figure 4–1 illustrates the relationship between user groups, users, and permissions.

Figure 4–1 Ownership Relationships Between User Groups, Users, and Permissions



User Groups

You can set up user groups with the following access rights:

- **Read Only**: Users can view all parts of IP Service Activator's object model but they cannot make any changes. Permissions cannot be set for a Read Only group.
- **Read Write**: Users can view all or selected parts of IP Service Activator's object model. Group permissions may be defined that restrict the actions that group members can perform.
- **Super User**: Users can view all parts of IP Service Activator's object model and perform unlimited actions. Only members of a Super User group can create new users and perform system configuration. Permissions cannot be set for a Super User group.

The access rights that are set for a Read Write user group apply to all members of the group.

Access rights associated with a Read Write user group can be fine tuned by setting detailed permissions on specific classes of object. These permissions may vary between Read Write groups. For example, one group may restrict its members to Read Only permissions for classes of service while another group may give access to classes of service but restrict access to VPN creation.

Note: User groups can be created only by users who have Super User access.

Users

Every user in IP Service Activator is a member of a single user group. The level of access a user has to IP Service Activator's commands depends on the group to which he or she belongs. Members of a group that has Read Write access are more restricted in the actions they can perform than members of a group that has Super User access. This difference in access level is reflected in the user interface. IP Service Activator elements to which a user does not have access are hidden from the user. For example, users with restricted access may be unable to view selected tabs and menu options.

It is possible to create a user account that can be used by several users concurrently. So, for example, you can create a guest account that can be used by multiple users to log in to IP Service Activator at the same time.

> **Note:** Users can be created only by users who have Super User access.
>
> There must always be at least one user who has Super User access. IP Service Activator enforces this rule by preventing deletion of the last user with Super User access.

Passwords

All users access IP Service Activator by username and password. Only users with Super User access can set up passwords.

A user can be forced to change his or her password at the next login to IP Service Activator and the password set to expire after a set number of days.

> **Note:** Password options are set in two different places.
>
> **Global password options** are set on the **User Passwords** property page of the **Options** dialog box. To access this dialog box, select **Tools**, and then select **Options** from the main menu of the **IP Service Activator** User Interface.
>
> See "Creating Rules for Passwords".
>
> **Password options for particular users** are set by accessing the **Password** property page of the **System User** dialog box for the user. To access this dialog box, right-click on the user in the **System User Groups** folder, and select **Properties** from the pop-up menu.

It is also possible to create rules for passwords. Users with Super User access can:

- Specify the minimum and maximum length for passwords.
- Set an expiry period that applies to all passwords.
- Specify password complexity.
- Specify the number of passwords that IP Service Activator remembers for each user.
- Specify the number of login attempts IP Service Activator allows before barring a user. A user with Super User access can reinstate barred users.

Users with Super User access are never barred after successive failed logins, but IP Service Activator records the number of failed logins in the system log file and displays an error message in the current faults pane.

> **Note:** Passwords and password rules can only be created by users who have Super User access.

Permissions

Permissions define the actions that can be performed on a class of objects or an instance of an object or the ability to perform an operation, such as discovering devices. Permissions can be set at the following levels:

- Group level: Super Users can set permissions for groups that have Read Write access rights.
- Object level: users with Read Write or Super User access can set permissions on the objects they own. Different permissions can be set for themselves, the group to which they belong and to members of other groups.

Additional attributes of permissions include:

- A permission that is set for a Read Write group applies to a class of object (object type). A permission that is set for a specific object applies to that object instance only.
- A user who assigns permissions to a previously unowned object assumes ownership of the object by default, or can assign ownership to another user.
- To calculate an object's actual permission for a user, the system combines the permission assigned to the object with the permission assigned to its class of objects, for that user.

Permission Levels

In the following descriptions, "class of objects" is the same as "object type". Also, "objects of the class" is the same as "instances of an object type". The following permission levels are available:

- Denied: the user cannot access the object or class of objects
- Read: the user can view the object or class of objects
- Link (Reference Only): objects or a class of objects can be linked to (referenced by) other objects, but cannot have other objects linked to them
- Link: the user can link the object/class of objects to appropriate objects, and appropriate objects can be linked to the object/class of objects
- Modify: the user can modify the object/class of object's properties
- Write: combines the Modify and Link permissions
- Create: the user has Write permission plus the user can create the class of objects
- Execute: the user can perform the operation

For a more detailed description, refer to the IP Service Activator online Help topic on object permissions. More detailed topics on Link, Link (Reference Only), and Execute are provided later in this section.

Table 4–1 summarizes how each permission level restricts the possible operations on objects:

Table 4–1 Permission Level Restriction Summary

Permission level	View	Link From	Link To	Unlink	Modify	Create	Delete	Own
Denied	N	N	N	N	N	N	N	N

Table 4–1 (Cont.) Permission Level Restriction Summary

Permission level	View	Link From	Link To	Unlink	Modify	Create	Delete	Own
Read Only	Y	N	N	N	N	N	N	N
Link (Reference Only)	Y	Y	N	N	N	N	N	N
Link	Y	Y	Y	Y	N	N	N	N
Modify	Y	N	N	N	Y*	N	N	N
Write	Y	Y	Y	Y	Y*	N	N	N
Create	Y	Y	Y	Y	Y	Y	Y	Y

*Except the **Ownership** property page cannot be modified.

Instances of most object types can be owned and permissions can be set, except for the following object types:

- User groups and users
- Some rule components – 802.1p User Priority, IP Protocols, Packet Markings and Traffic Types
- Device types
- System logs
- The policy server

Permission Examples

- A user who has been granted Read Only permission on a VPN can view the VPN but cannot link other objects to the VPN or unlink objects from it, modify, create, delete, or own the VPN.
- A user who is a member of a Read Write group that has Modify permission on the policy area of the object model can view classes of service and modify their property values. However, the user cannot link the CoS to other objects, unlink existing associations, create, delete, or own classes of service.

Link Permission

Link permission refers to linking the target object to another object and also to linking another object to the target object. This means, for example, that if you have link access for a domain or a site, you can link to these objects without necessarily having permission to modify them.

Users can view objects of the selected class and folders and tabs that contain objects of the class. Objects of the class can be linked to other objects and can have other objects linked to them. Property pages are read-only. Options to create objects of the class are not shown and options to delete objects with this permission are disabled.

Link (Reference Only) Permission

Link (Reference Only) is a more restrictive version of the Link permission:

- Users can view objects of the selected class and folders and tabs that contain objects of the class.

- Objects or classes of objects can be linked to (referenced by) other objects, but cannot have other objects linked to them. The referenced object can be unlinked from the other objects.
- Property pages are read-only. Options to create objects of the class are not shown and options to delete objects with this permission are disabled.

When Link (Reference Only) is assigned to a parent object, this permission is not automatically enforced to all children in the hierarchy. The user is responsible for setting and controlling permissions down the hierarchy of objects.

On Ownership property pages of classification groups and compound traffic objects, the **Objects below this level inherit the permissions** check box does not apply. The children of these objects do not inherit their parents' permission level. This restriction prevents unintended overwriting of the permission on an object [with multiple parents] that inherits different permissions from each parent.

Primary Purpose of Link (Reference Only) Permission

The unidirectional linkage of this permission maintains the definition of a policy object while using it to create new policy objects. In a common application, users need to use globally defined policy objects to define customer-specific policy data. Link (Reference Only) permission allows users to create a new policy object by referring to an existing policy object, but prevents accidentally changing the existing policy object that could result from linking another object to it.

Follow the recommended usage guidelines to ensure secure access to classification group and traffic type elements.

Usage Guidelines for Classification Groups

Use these Link (Reference Only) permission guidelines to secure the definition of your classification group and classification objects in the GUI:

- Create two folders, Folder1 and Folder2.
- Set Folder1 permission to Link (Reference Only) and select its "inherit" check box. (The actual check box label is "Objects below this level inherit the restrictions".)
- Move all existing top-level classification groups under Folder1.
- Set Folder2 permission to Denied.
- Under Folder2, create subfolder Subfolder2 with Link (Reference Only) permission and select its **Inherit** check box.
- For each top-level classification group contained in Folder1, link every element of the classification group into Subfolder2. (That is, link every classification, sub-classification group, and sub-classification.)

These guidelines ensure that all elements in any top-level Link (Reference Only) classification group maintain the same permissions. Because inheritance is not supported within classification groups, other users will be unable to overwrite these permissions.

Note: A Link (Reference Only) classification or classification group that either does not have a parent, or does not have a parent with restricted permissions, is in danger of being unlinked from its original location. Placing the classification and classification group objects within restricted folders (according to the above guidelines) provides a secure location for the objects.

Usage Guidelines for Traffic Types

Use these Link (Reference Only) permission guidelines to secure the definition of your traffic type policy objects in the GUI. Traffic Groups behave like traffic folders.

- Create 2 traffic groups: Group1 and Group2.
- Set Group1 permission to Link (Reference Only) and select its **Inherit** check box. (The actual check box label is **Objects below this level inherit the restrictions**.)
- Move all existing top-level traffic types and compound traffic under Group1.
- Set Group2 permission to Denied.
- Under Group2, create subgroup SubGroup2 with Link (Reference Only) permission and select its "inherit" check box.
- For each compound traffic object in Group1, link every traffic type contained in the compound traffic object into SubGroup2 (if the traffic type does not already exist directly under Group1).

These guidelines ensure that all traffic types in any top-level Link (Reference Only) compound traffic object maintain the same permissions.

Execute Permission

The Execute permission level can only be set for Read Write groups and applies to:

- Loading configuration data (policy files)
- Accessing a device via Telnet
- Managing a device
- Discovering the network
- Validating data
- Creating site accounts
- Finding system components
- Saving, merging and committing user transactions

Folder Permissions

- The object permissions assigned to a folder are inherited down to its sub-folders when the **Inherit** flag (**Objects below this level inherit the restrictions** check box) is selected. The inherited permission can be overwritten by assigning independent ownership to a sub-folder.
- Permissions assigned to a Policy object folder are inherited by the top-level policy objects contained in that folder when the inherit flag is selected.

> **Note:** Folder permissions are not inherited by classifications contained in a classification group within the folder. Similarly, the permission assigned to a traffic group is not inherited by traffic types contained in a traffic compound within the traffic group.

- An object in a folder cannot be moved to another folder unless the user has Create permissions for the object, or the object is un-owned.

Impact of Permissions in the GUI View

The user interface displays information relevant to the permissions that apply to the user group.

If access to an object class (object type) or instance of an object is denied, the object and any folders and tabs that exclusively contain that object are not displayed to the relevant users. In addition, objects do not appear on property pages, menus, and drop-down menus.

Multiple User Interfaces

Because changes to permissions only take place the next time that a user logs in to IP Service Activator, it is possible for different permissions, and thus different views of the system, to apply to members of the same user group on two user interfaces.

Note: Permissions set for Read Write groups are calculated and fixed for the duration of a user's login session. Changes to group permissions are not apparent to the group's members until the next time they log into IP Service Activator.

Permissions set for an instance of an object are apparent to other users immediately. For information on owning and setting permissions on an object, see "Owning and Setting Permissions on an Object".

We recommend that a system Super User makes changes to permissions only when no other users are logged in. If necessary, the Super User should disable access to the relevant user group while changes are being made (see "Disabling or Re-enabling User Access").

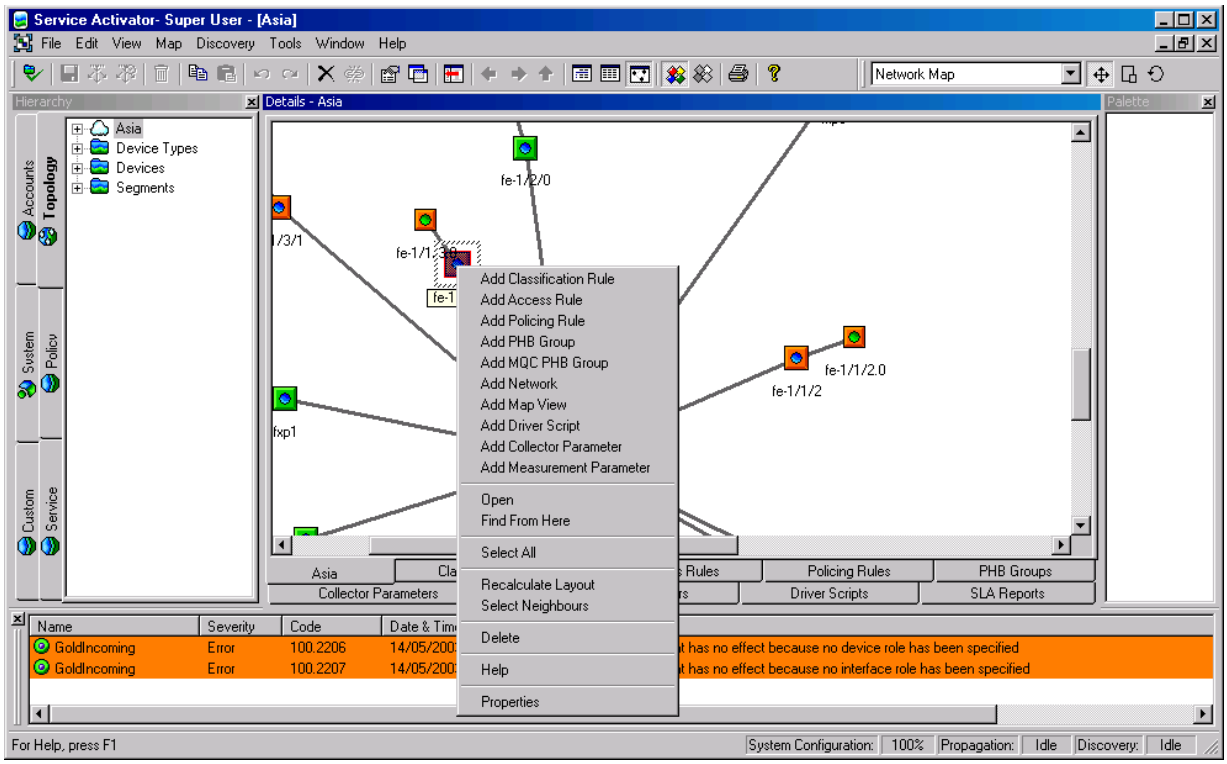
Inheritance of Permissions on Faults

Faults inherit object permissions from the object on which the fault is reported. This ensures that users do not see faults that are reported on objects that they do not have access to.

GUI Example

For a Read Write user who has access to all parts of the system, the IP Service Activator user interface appears as shown in Figure 4–2.

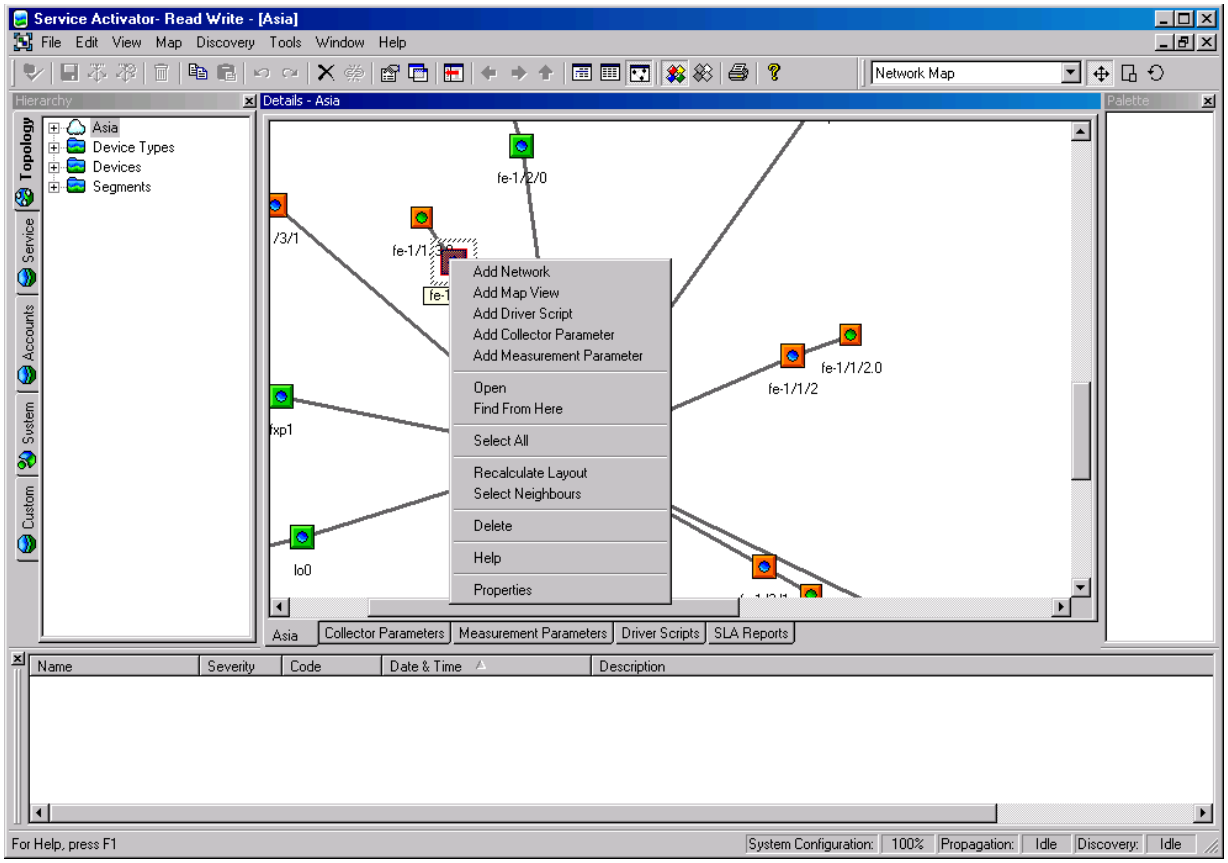
Figure 4–2 IP Service Activator GUI: Topology View with Full User Access



All tabs are visible and options to add policy and measurement elements and driver scripts are available on the pop-up menu.

For a user who has Read Write access with access denied to policy objects, the user interface appears as shown in Figure 4–3:

Figure 4–3 IP Service Activator GUI: Limited User Access Topology View



The **Policy** tab is not displayed and the options to create PHB groups and rules are not available on the pop-up menu.

Also that you cannot see faults associated with object types to which you do not have access. In this example, the warnings that relate to a PHB group with no roles associated with it cannot be seen in the second illustration.

Inheritance of Permissions

The permissions set on a class of object for a group or on a specific object by its owner can be inherited by lower level objects. For example, applying Read permission at device level can be inherited so that a group's users also have Read permission for interfaces, sub-interfaces, and VC endpoints.

The IP Service Activator permission hierarchy is divided into three main areas: Custom, Domain, and System.

Custom

Within the Custom area, inheritance applies to driver scripts and script context.

Inheritance of Permissions on Sites and VPNs

Take care when setting up permissions for sites and VPNs to avoid sites being hidden from users. This can occur when sites that are a child of one customer are linked to multiple VPNs with different security permissions.

- When sites that are already included in a VPN are subsequently added to a management VPN, to ensure that sites in the management VPN can be seen by the

customer when logged in, do not select the **Objects below this level inherit the restrictions** check box on the **Ownership** property page of the management VPN object.

- When sites owned by one customer are included in a VPN owned by another customer, to ensure that sites can be seen by any other customer, ensure that **Access for other users** for all Site objects is set to at least Read.

Changing the Default User

When the policy server is started for the first time after installation, it creates a default user with Super User access rights. Any user interface components that are subsequently started automatically use the default user's username and password details to provide access – the IP Service Activator main screen appears immediately and no username and password details are required.

We strongly recommend you change the default user group and default super user as soon as possible. After you change the default user, all users must enter a name and password each time they start the IP Service Activator user interface.

> **Note:** For complete dialog box and property page descriptions, refer to IP Service Activator online Help.

To change the default user:

1. Select the **System User Groups** folder on the **Systems** tab in a global setup or domain management window.
2. Select the **Super Users** group. Initially there is a single user named *admin*.
3. Select the *admin* user, then select **Properties** from the pop-up menu. The System User dialog box opens.
4. On the **System User** dialog box, **System User** property page, change the **Username** field to an appropriate unique entry.
5. Ensure that the **Allow concurrent logins** check box is selected.

> **Note:** Enabling the **Allow concurrent logins** option will allow you to log in to IP Service Activator using the same account from multiple user interfaces. As well, this option is required for some of IP Service Activator's optional modules to perform automated logins.

6. Select the **Password** property page.
7. Change the **Password** to an appropriate unique password, and confirm the password by re-entering it in the **Confirm Password** field.
8. Click **OK** and commit the transaction.

We recommend that you set up additional users with different levels of access. To do this, you must first create additional user groups with the relevant access levels. You can then add users to these groups. For more information, see "Setting Up User Groups" and "Creating Users".

Creating Rules for Passwords

Users with Super User access can set rules for passwords, such as the minimum and maximum length a password must be and the period after which passwords expire.

Note: Password options can also be set for an individual user account.

When a password expires, the user is prompted for a new password on the next login to IP Service Activator's user interface.

To edit global password rules:

1. From the **Tools** menu, select **Options**.

   The Options dialog box opens.

2. Select the **User Passwords property** page.

3. Enter values for the password, including **Minimum Length**, **Maximum Length**, P**assword Complexity**, **Password Re-use**, **Passwords expire after**, and **Disable account after login failures**.

4. Click **OK**.

For complete dialog box and property page descriptions, refer to IP Service Activator online Help.

Setting Up User Groups and Users

Users with Super User access can define access to the system by setting up new user groups and user logins.

Setting Up User Groups

A user group defines the access level that its members have to IP Service Activator. Every user is a member of a user group.

IP Service Activator automatically creates one user group on installation. The group has Super User access and contains a single user (see "Changing the Default User"). To create users with Read Write and Read Only access, you must set up additional user groups. For Read Write groups, you can also define the permissions that apply to the

group – that is, which parts of IP Service Activator the group's members can view and/or modify.

Some example Read Write user groups might include:

- Network engineers: group members can import topology files, define roles, create maps and create driver scripts
- Policy engineers: group members can create traffic types and PHB groups
- Customer service engineers: group members can manage customers and their associated VPN services
- Demonstration users: group members can make changes but cannot save them

A user can only be a member of a single group.

To set up a user group:

1. Select the **System User Groups** folder on the **Systems** tab in a global setup or domain management window.
2. Select **Add System User Group** from the pop-up menu.

   The System User Group dialog box opens.
3. On the **System User Group** property page specify details including **Name**, **Access Rights**, and **Remarks**, **Disable all group users**. Under **Access Rights** you can select the **Read Write**, **Read Only**, or **Super User** option.
4. Click **OK**.

For complete dialog box and property page descriptions, refer to IP Service Activator online Help. Also, for more information about permissions, see "Setting Up Read Write Group Permissions".

Setting Up Read Write Group Permissions

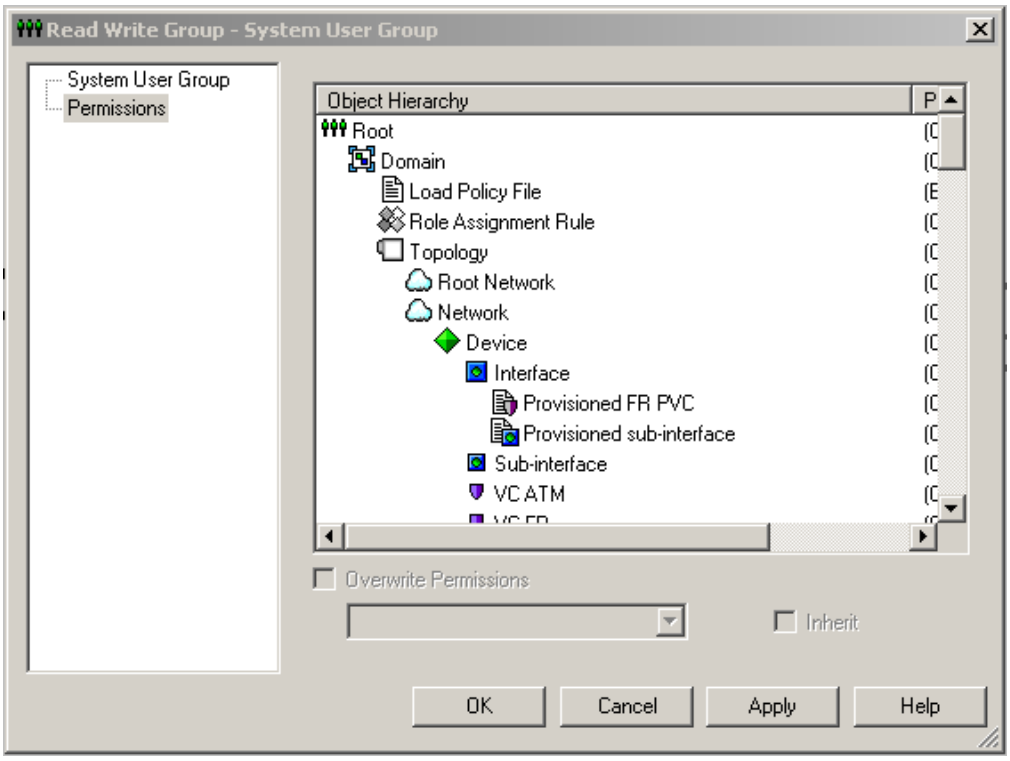
For Read Write user groups, group permissions define what actions members of the group can perform on classes of object. Group permissions allow groups to be restricted according to the roles and responsibilities of the group's members.

Users with Read Write or Super User access can also own and set permissions on a specific instance of an object. The permissions set on a specific object may override those that apply to a class of objects for a group. For more information, see "Owning and Setting Permissions on an Object".

To set or change a permission for a user group:

1. On the **System** tab, open the **System User Group** folder and select **Properties** from the relevant user group's pop-up menu.

   The System User Group dialog box opens.
2. Select the **Permissions property** page.



Group permissions are displayed in a fixed hierarchy, and the appropriate permissions are displayed for each level within the hierarchy. For more information about permissions and the permissions hierarchy, see "Permissions".

1. Select the object class whose permissions you want to set.
2. Click the **Overwrite Permissions** check box, and then select the permission level you require from the drop-down menu.
3. (Optional) If you want object classes at a lower level in the hierarchy to inherit the permission, select the **Inherit** check box.
4. Click **OK**.

Note: You can select individual object classes at a lower level within the hierarchy and change the individual permissions as required.

When you change permissions for a user group, the changes are only apparent to group members the next time they log into IP Service Activator.

Creating Users

Users with Super User access can set up new users, specifying their username and password details. It is possible to force users to change their password when they first log into IP Service Activator.

A user must be set up within a user group, and can be a member of only one group.

Note: For complete dialog box and property page descriptions, refer to IP Service Activator online Help.

To set up a user:

1. Select the **System User Groups** folder on the **Systems** tab in a global setup or domain management window.
2. Select the group to which you want to add a new user.
3. Select **Add System User** from the pop-up menu.

   The System User dialog box opens.
4. On the **System User** property page, enter details including **Username**, **Remarks** and **Allow concurrent logins**.

   See "Changing the Default User" for more information on **Allow concurrent logins**.
5. Select the **Password** property page.
6. Enter details including **Password**, **Confirm Password**, **User must change password at next login**, and **Expires after**.
7. Click **OK**.

For complete dialog box and property page descriptions, refer to IP Service Activator online Help.

Resetting a user password

If a user with Super User access changes a user password, the user must enter this new password on re-entry to IP Service Activator.

Disabling or Re-enabling User Access

You can disable access for all members of a group or for an individual user. Disabling access for a user who is logged in forces the user out of IP Service Activator.

To disable access for all members of a group:

1. On the **Systems** tab in a global setup or domain management window, select the **System User Groups** folder.
2. Click the relevant group.
3. Select **Properties** from the pop-up menu.

   The System User Group dialog box opens.
4. On the **System User Group** property page, select the **Disable all group users** check box.
5. Click **OK**.

To disable access for an individual user:

1. Select the **System User Groups** folder on the **Systems** tab in a global setup or domain management window.
2. Select the group that contains the user whose access you want to disable, then click the relevant user.
3. Select **Properties** from the pop-up menu.

   The System User dialog box opens.
4. On the **System User** property page, select the **Disable user** check box.
5. Click **OK**.

For complete dialog box and property page descriptions, refer to IP Service Activator online Help.

Re-enabling Users

If a user is locked out of IP Service Activator as the result of too many failed login attempts, a user with Super User access can re-enable the user by deselecting the **Disable user** check box on the user's properties pages. If necessary, you can increase the number of permitted login attempts; for more information, see "Creating Rules for Passwords".

Viewing User Group and User Information

You can view user group and user information and see which users are logged into IP Service Activator.

Users with Super User access can see all user group and user information. Users with Read Write or Read Only access can only view the group to which they belong and their own user details.

To view user group information:

- Double-click on the **System User Groups** folder.

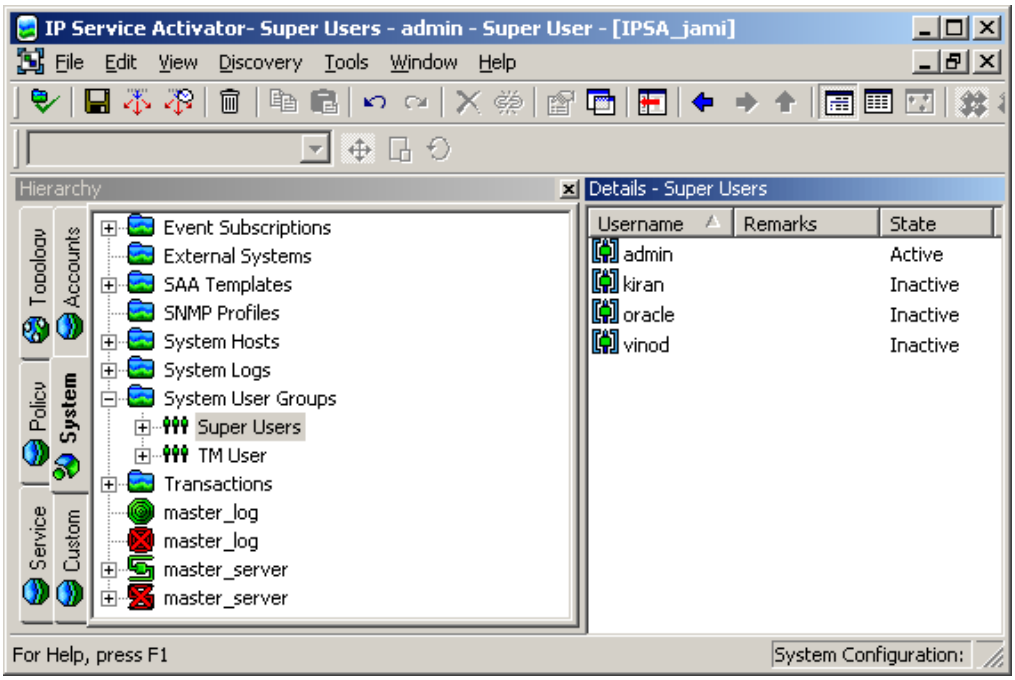
The information is displayed in the **Details** pane as follows:

- **Name**: The name of the user group.
- **Access Rights**: The access rights of the user group
- **Remarks**: Any remarks that were entered when the user group was set up.
- **Active Users**: For each user group, lists the number of users currently logged in to IP Service Activator.

To view user group member details:

- Open the **System User Groups** folder and double-click the user group whose membership details you want to view.

  The information is displayed in the **Details** pane:



The details are as follows:

- **Username**: The name of the user.
- **Remarks**: Remarks that have been added about the user.
- **State**: The state of the user:
  - **Inactive**: the user is enabled with no active sessions.
  - **Active**: the user has active sessions (if more than one session is active, the number of sessions appears in parentheses)
  - **Disabled**: the user's access has been disabled by a Super User
  - **Denied**: the user has been denied access because the prescribed number of login attempts was exceeded

---

Note: You can also view the status of a user by selecting the required user, then select **Properties** from the pop-up menu. The status field shows the current status of the user.

Allowing a User Group to Access Configuration Template Module

You can grant users access to Configuration Template Module (CTM) by assigning permissions to existing IP Service Activator user groups for a device, domain or network. It is recommended to restrict access to CTM functionality for some users based on their level of authorization.

Before you can allow users to access CTM, you must have already created user groups. See "Setting Up User Groups" for more information.

To allow a user group to access CTM:

1. In the IP Service Activator GUI, right-click the name of the device, domain, or network.

2. From the drop-down menu, select **Modules**, then **Configuration Template**, then **Group Admin.**

   The CTM Group Permissions dialog box appears.

3. Select the correct group name.

4. Select any of the following check boxes:

   - **Activation**: the user group can access the Activation GUI and use a template to provision a device
   - **Admin**: the user group can access the Administration GUI to create and edit templates
   - **Audit**: the user group can access the Audit GUI to review template audit history

   Note: The Super User group has all permissions by default. You cannot change Super User permissions in the CTM Group Permissions dialog box.

5. Click **OK**.

For more information about using CTM, see the IP Service Activator online Help topic about managing CTM user groups.

Owning and Setting Permissions on an Object

If you create an object, you can assume ownership of that object and specify which users can view and modify the object. You can set different permissions for yourself, other members of your user group and members of other groups. Ownership of an object can also be passed to another user.

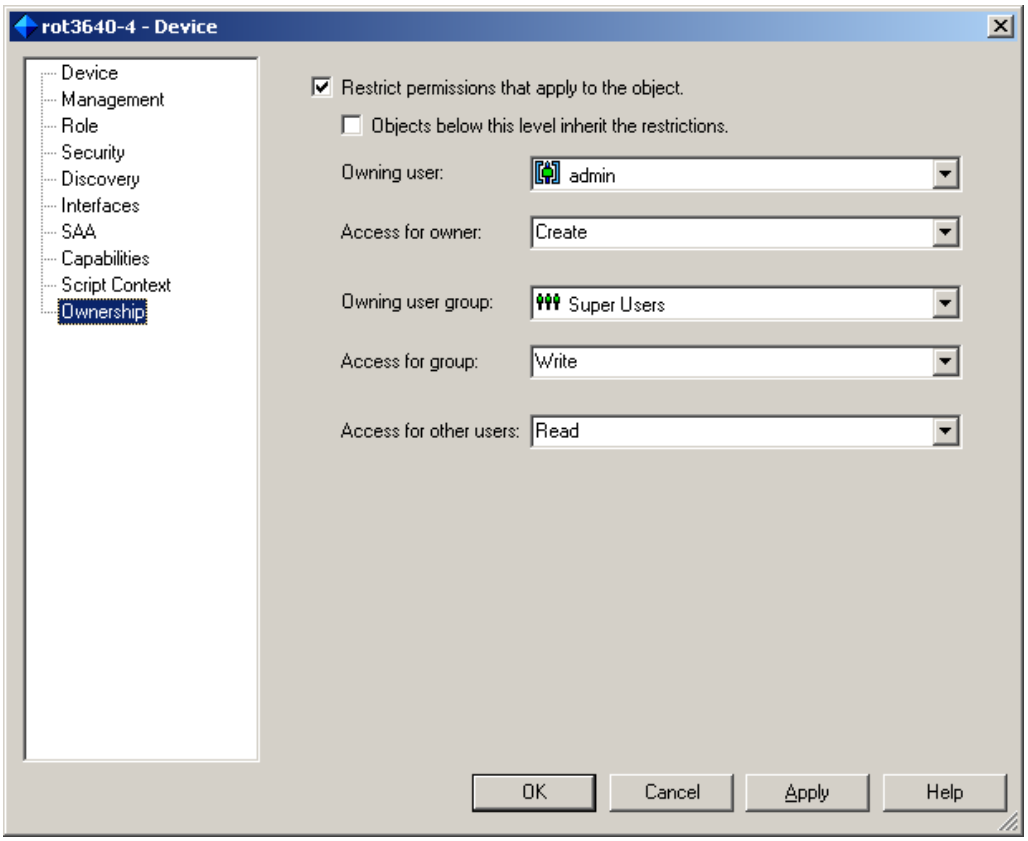
Permissions can be set on most objects except user groups and users, some rule components, device types, system logs, and the policy server. For further information about permissions refer to "Permissions".

To set permissions on an object:

1. Select the object you want to set permissions on, then select **Properties** from the pop-up menu.

   The object's dialog box opens.

2. Select the **Ownership** property page:



3. Enter details to specify permissions, ownership, and access. For complete dialog box and property page descriptions, refer to the **Ownership Pages** topic in IP Service Activator online Help.
4. Click **OK**.

Note: If a user is deleted, any objects owned by that user are left ownerless.

A user who assigns permissions to an unowned object assumes ownership of the object by default.

Ownership of an object can be changed by any user with Create permission on the object.

Running the User Interface in Confirmed Commit Mode

IP Service Activator uses a transaction-based model for updating the system and implementing configuration changes.

By default, a transaction that has been committed in the user interface is queued and processed by the policy server as a background task. This means that a user can continue working in the user interface and commit further transactions while the policy server saves data to the database. However, if the policy server fails any transactions that it had not yet written to the database will be lost.

In confirmed commit mode, a user who has committed a transaction can only make changes in the user interface when the transaction has been successfully saved to the database.

Note: If your deployment integrates IP Service Activator with other OSS systems, it is particularly important that these systems are guaranteed persistence of data. This ensures synchronization between systems. You may therefore wish to consider running IP Service Activator user interfaces in confirmed commit mode in this scenario.

To run a user interface in confirmed commit mode:

1. From the **Tools** menu, select **Options**.

   The Options dialog box opens.

2. Select the **Transactions** tab and select the **Confirmed transaction mode** check box.
3. Click **OK** or **Apply**.

Context Menu Extension Plugin

The Context Menu Extension Plugin lets you add new entries in the context menu, to launch external applications from the GUI. Additionally, the Context Menu Extension Plugin is used to supply IP Service Activator login attributes to the external application. This avoids the need to re-enter these attributes when the external application needs to communicate with IP Service Activator.

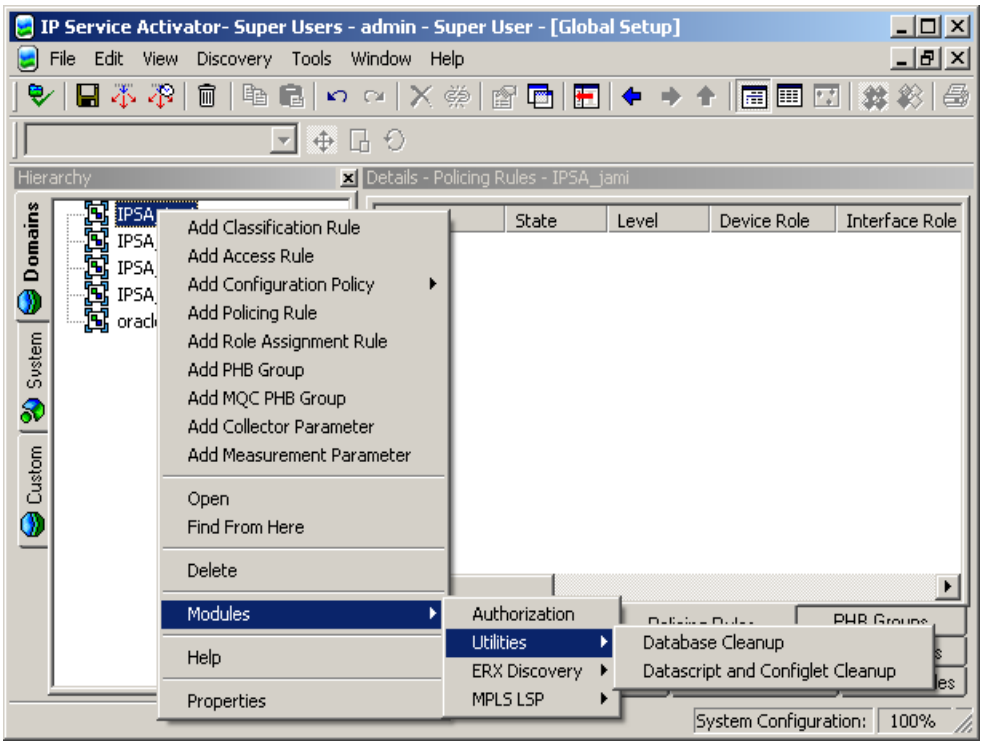
External applications can be written in any scripting language, including Perl, Python, or a Unix shell script. The external application can be applied only to the objects defined for it in the **ExplorerScripts.xml** file.

Context Menu

The context menu is the drop-down menu that appears when you right-click an object in the IP Service Activator GUI. A menu item appears in the Modules sub-menu when it is defined in the **ExplorerScripts.xml** file.

Figure 4–4 shows the context menu for the Domain object, as defined in the example **ExplorerScripts.xml** file provided later in this section.

Figure 4–4 Context Menu for Domain Object



Examples of existing context menu extensions include the Module Authorization GUI, SSH Launch, Database Cleanup, and Datascript Cleanup.

Setting up the Context Menu Extension Plugin

The Context Menu Extension Plugin is installed with the Common module. Refer to *IP Service Activator Installation Guide* for the Common module installation procedure.

Creating Context Menu Items in the GUI

Use this procedure to add new menu items in the context menu drop-down list, in the Modules sub-menu. New menu items are implemented by modifying the **ExplorerScripts.xml** file.

To create new menu items

1. Edit the *Service_Activator_Home*/**ExplorerScripts/ExplorerScripts.xml** file to add a new menu entry. (Examples and notes follow this procedure.)
2. Log out of the GUI and log back in to view the new menu item.

Example ExplorerScripts.xml File

The **ExplorerScripts.xml** file defines which applications you can launch from the context menu, the name of the menu item, and the object types on which each application will operate. An excerpt from the **ExplorerScripts.xml** file follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE IpsaExplorerScripts [
<some definitions omitted here>
<!ELEMENT IpsaExplorerScripts (ScriptMenu,ObjectTypeList)>
]>
```

```
<IpsaExplorerScripts>
<!--List of scripts in main menu-insert a ScriptSubmenu tag for submenus-->
<!--Each submenu must have a name (which appears in the menu) -->
<!--Each script must have a name (which appears in the menu), command line->
<!--and list of affected OM object types - available types appear below-->

<ScriptMenu>
<Script command="&quot;path; eeucommon.jar
     com.mslv.osa.custom.executable.LaunchExec %1 %2 %3 %4 %5 start ssh.exe"
     name="SSH" objects="Device"></Script>
<Script command="&quot;path; eeucommon.jar
     com.mslv.osa.custom.moduleauth.Authorization %1 %2 %3 %4 %5"
     name="Authorization" objects="Domain"></Script>
<ScriptSubmenu name="Utilities">
<Script command="&quot;path; eeucommon.jar
     com.mslv.osa.gui.DatabaseCleanup %1 %2 %3 %4 %5 Domain"
     name="Database Cleanup" objects="Domain"></Script>
<Script command="&quot;path; eeucommon.jar
     com.mslv.osa.gui.DatascriptCleanup %1 %2 %3 %4 %5 VPN"
     name="Datascript Cleanup" objects="VPN"></Script>
<Script command="&quot;path; eeucommon.jar com.mslv.osa.gui.DatascriptCleanup %1
%2 %3 %4 %5 Customer"
     name="Datascript Cleanup" objects="Customer"></Script>
</ScriptSubmenu>
</ScriptMenu>
</IpsaExplorerScripts>
```

In this example, **path** is used instead of the following full path to the launcher file:

C:\Program Files\Oracle Communications\Service Activator\modules\bin\launcher.bat

Resulting Context Menus

As indicated in Table 4–2, the menus resulting from this example file are context-specific to the object that is right-clicked. The screen illustration in "Context Menu Extension Plugin" shows the context menu for the Domain object.

Table 4–2 Description of Resulting Context-Specific Menus

When you right click	You see this menu
a device object	Modules -> SSH
a domain object	Modules -> Authorization Modules -> Utilities -> Database Cleanup
a VPN object	Modules -> Utilities -> Datascript Cleanup
a Customer object	Modules -> Utilities -> Datascript Cleanup

A menu item is visible in the context menu only if you right-clicked an object for which the menu item has been defined.

Sample Command Line Explained

```
<Script command="&quot;path; eeucommon.jar
     com.mslv.osa.custom.executable.LaunchExec %1 %2 %3 %4 %5
     start ssh.exe" name="SSH" objects="Device"></Script>
```

where

- command=""path; eeucommon.jar

  com.mslv.osa.custom.moduleauth.Authorization %1 %2 %3 %4 %5"

  defines the external application (ssh.exe) and its parameters

- **%1**: Object ID (ID of the object that was right-clicked in the GUI)

  %2: UserID **%3**: Password **%4**: Host **%5**: Port

  (Values for these login attributes are passed to the external application, permitting it to communicate with IP Service Activator without requiring you to log in again.)

- name="SSH" defines the name of the item that appears in the menu
- objects="Device" defines the object context (the object types for which, when right-clicked, this menu item appears). You can add more object types inside the quotation marks, with object types separated by a space. For example: objects="Device Interface Sub-interface"

  Alternatively, you can replicate the command line for each object type.

  The real **ExplorerScripts.xml** file lists the full set of objects to which menu items can be applied.

Notes on Creating Context Menu Items by Editing ExplorerScripts.xml

- Using the syntax described, a menu item is added even if the external application is disabled. For a menu item selection to work, ensure that its external application is properly enabled.
- The sequence of items in the context menu is the same as the lines in the **ExplorerScripts.xml** file. To add a new menu entry, edit the section of the file between tags **<ScriptMenu>** and **</ScriptMenu>**.
- The example file also shows creation of sub-menu **Utilities**. Sub-menu entries are entered between tags **<ScriptSubmenu name="Utilities">** and **</ScriptSubmenu>**.

The Module Authorization GUI

When using IP Service Activator modules, a separate Module Authorization GUI is used to control user group access to certain module functions. The Module Authorization GUI is used with the following modules:

- Common module
- MPLS LSP module

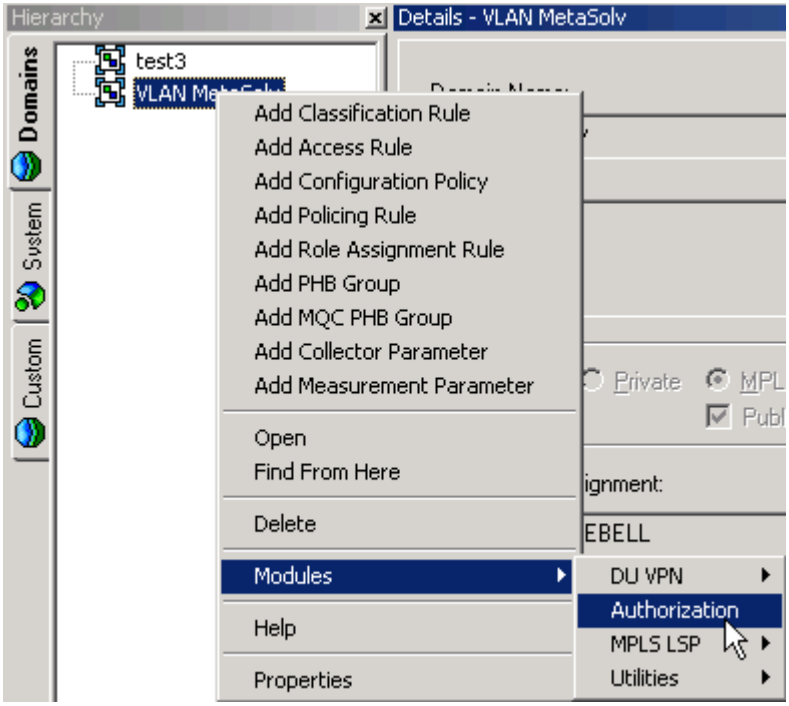
The Module Authorization GUI is installed automatically with the Common Module.

Accessing the Module Authorization GUI

By default the Module Authorization GUI can only be accessed by users in the group Super Users. Users in the Super Users group are automatically granted access to all modules and are not restricted by settings in the Module Authorization GUI.

To access the Module Authorization GUI:

1. Right-click the Global **Domain** object for which you want to configure module permissions.



2. From the pop-up menu, select **Modules**, and then select **Authorization**.

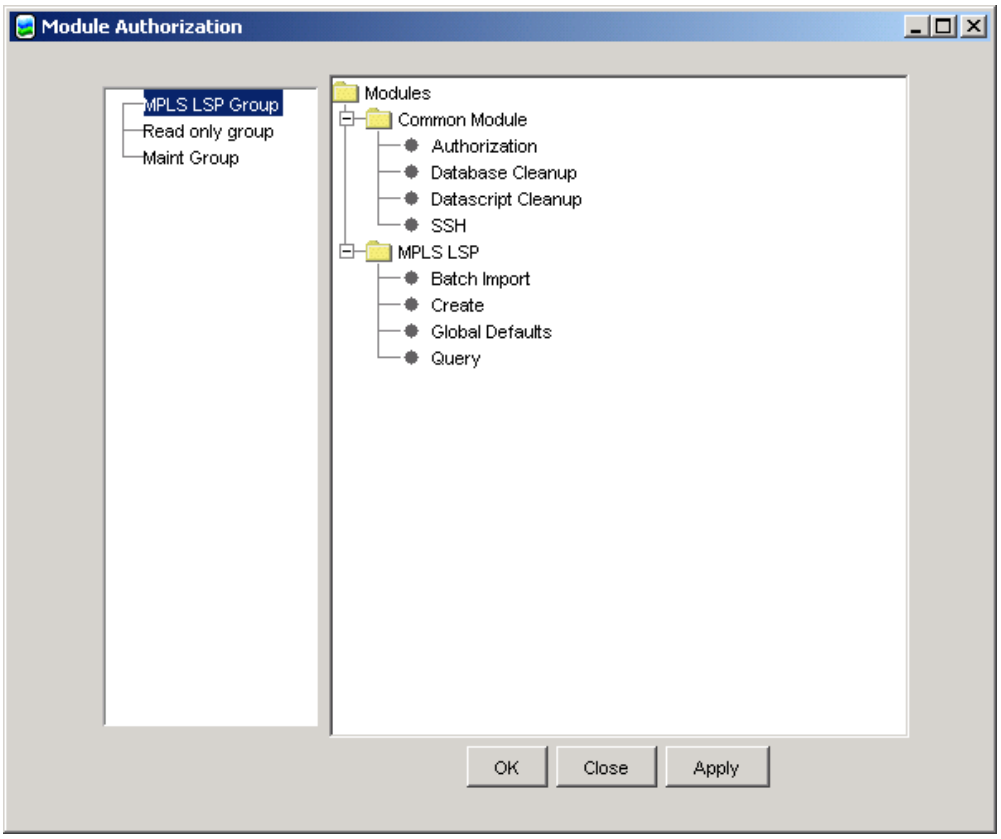
   The Module Authorization GUI is displayed.

Setting Permissions with the Module Authorization GUI

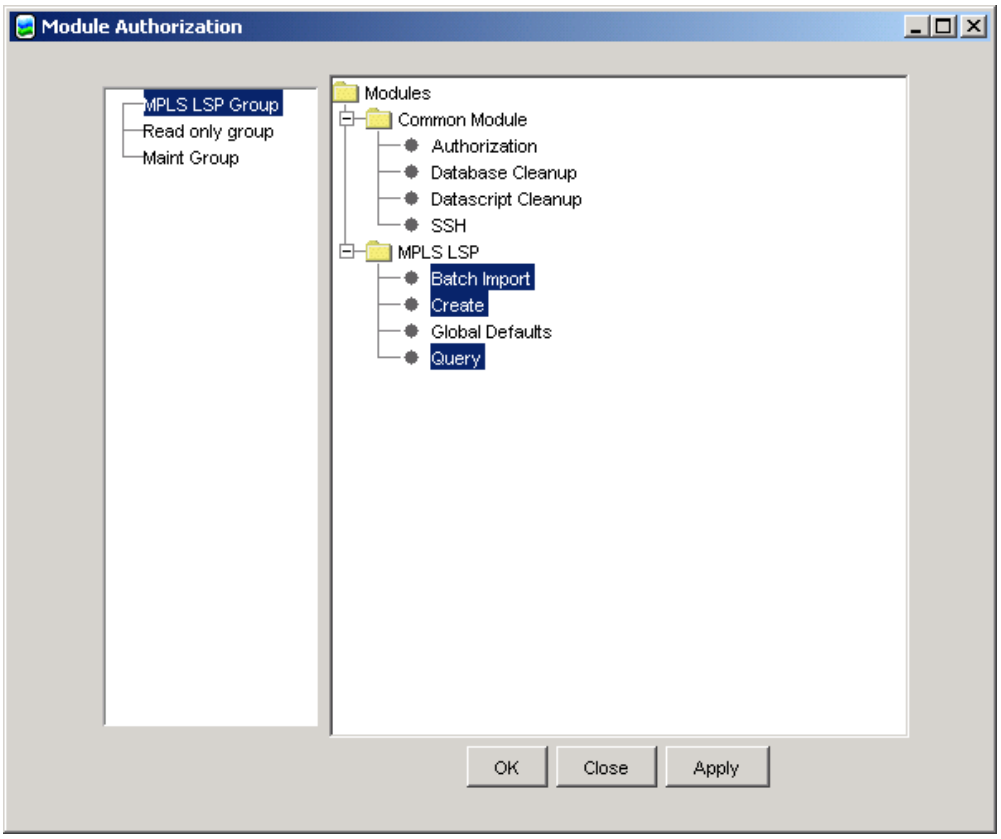
To change permissions for a particular user group, select it from the list in the left panel.

Hold the Ctrl key and click permissions to add or remove them for that group. When a permission is highlighted it is selected for the group.

In the following example, the MPLS LSP Group has access to none of the available functional areas of either the Common Module, or the MPLS LSP module.



After selecting access to the Batch Import, Create and Query functions for the group (but not the Global Defaults function), the GUI looks like this:



The Module Authorization GUI has three buttons.

- Click **OK** to commit changes and exit
- Click **Apply** to commit your changes but leave the GUI open
- Click **Close** to close the GUI. If you have made any changes that are not saved, you are prompted to save them, discard them, or return to the GUI.

5

Backing Up and Restoring Data

This chapter describes how to back up and restore Oracle Communications IP Service Activator data on Windows, Solaris, and Linux.

Introduction

Regular back up of IP Service Activator data is essential and we strongly recommend that you perform a routine daily back up, perhaps as an overnight process. A check on the integrity of the database is strongly recommended, and this check must be performed on the backup database, rather than the live system. For information on checking the integrity of the database, see "Managing the IP Service Activator Database".

> **Note:** IP Service Activator supports multiple platforms like Windows, Solaris, and Linux.

IP Service Activator Installation Directories

The default installation directory for IP Service Activator server software on Solaris/Linux is: **/opt/OracleCommunications/ServiceActivator**.

> **Note:** This chapter refers to the IP Service Activator installation directory as *Service_Activator_Home*.

The default installation directory for IP Service Activator user interface software on Windows workstations is: **Program Files\Oracle Communications\Service Activator**.

Data to be Backed Up

This section provides an overview of the critical and non-critical data for back up:

- Critical system data must be backed up on a regular basis in order to ensure that the system can be restored to a known functioning position in case of a problem.
- Non-critical data is not essential to the running of IP Service Activator but should be backed up if you want to restore the system exactly to its previous state.

Critical Data

- The Oracle database tables

- **Component Names**: during installation, system components are assigned names automatically by the installation program using the following convention:

  component_type-host_name

  If the proxy agent needs to be reinstalled it should be given the same name, otherwise all existing device associations will be lost – that is, the system will not be able to associate the proxy agent with previously-allocated devices.

  – On Solaris/Linux, component names are stored in *Service_Activator_Home*/**Config/cman.cfg**

- **Component Manager settings**: these are configured on installation. If these are corrupted, the system on the relevant host must be reinstalled. The location of these settings depends on the operating system:

  – On Solaris/Linux, component manager settings are defined in *Service_Activator_Home*/**Config/cman.cfg**

- **Naming service settings**: these define the naming service host machine name and the port number to be used for communication:

  – On Solaris/Linux, these are defined in *Service_Activator_Home*/**Config/omniorb.cfg**

Non-critical Data

- **User Interface settings**: there are user preferences that may affect column ordering selections, and so on.

  – On Windows, user interface settings are defined beneath the **HKEY_CURRENT_USER\Software\OracleCommunications** registry key.

- **Network Processor audit logs**: log files for each device, which are cycled once a day. These are stored in the **Audit Trails** subdirectory beneath *Service_Activator_Home*.

- **Installed Components**: the installed components and related files on the host. These are stored in the *Service_Activator_Home*/**bin** subdirectory on Solaris/Linux hosts.

Backing Up and Restoring Data on Windows

This section describes how to back up and restore IP Service Activator data on Windows workstations. For more information, see "Backing Up and Restoring Windows Registry Settings."

Full Restore

Restore each workstation where IP Service Activator software was installed. For more information, see "Backing Up and Restoring Windows Registry Settings."

Restart the system after the restore is complete.

Replacing the User Interface

This section describes how to replace the user interface software when the hardware or operating system has been replaced or reinstalled. You may need to do this if a host on which a system component is installed stops functioning and has to be replaced.

Note: When reinstalling components, you must enter the same serial number as was entered for the original installation.

For each workstation where a User Interface has been installed:

1. Reinstall the User Interface.

   You must enter the same serial number during installation as was entered for the original installation.

2. Restore the Registry settings **HKEY_LOCAL_MACHINE\SOFTWARE\OracleCommunications**. For more information, see "Backing Up and Restoring Windows Registry Settings."

Backing Up and Restoring Windows Registry Settings

This section explains how to back up and restore Windows Registry settings using regedt32.

To back up:

1. Start regedt32.
2. Select the appropriate key. For example: **HKEY_CURRENT_USER\SOFTWARE\OracleCommunications**.
3. Select **Export**.
4. Specify a filename.
5. Click **Save**.

To restore:

1. Start regedt32.
2. Select **Import**.
3. Select the file to be restored.
4. Click **Open.**
5. Ensure the key is restored to the correct place in the registry.

Backup and Restore on Solaris/Linux

This section describes how to back up and restore IP Service Activator data on Solaris/Linux.

Backup Procedures on Solaris/Linux

This section details the procedures required to back up the system.

Full Backup

Do a full backup:

- For the system as a whole: Back up the Oracle database (for details see the backup and Recovery procedures that come with your Oracle Database).
- For each host where IP Service Activator software is installed: Back up the *Service_Activator_Home* directory including all subdirectories.

Backup must include everything that appears under *Service_Activator_Home*, not just the data that physically lives there. For example, if you realize that you need more space for WorkingData and decide to mount additional resource (or symlink it to some other file system on the machine), that information must be included in the backup.

The *Service_Activator_Home* directory contains symbolic links to files. We recommend you use separate tar commands to back up, and running tar as the root user. For more information about the IP Service Activator directory structure, see the *IP Service Activator Installation Guide*.

Replacing One or More Components

This section details the procedures to replace components on a host when the hardware or operating system has been replaced or reinstalled. You may need to do this if a host on which a system component is installed stops functioning and has to be replaced.

> **Note:** When reinstalling components, you must enter the same serial number as was entered for the original installation.

For the host where the Policy Server software has been installed:

1. Reinstall the Common components, the Naming Service, the Policy Server, the Component Manager, and the System Logger.
2. Restore the **/opt/OracleCommunications/ServiceActivator/WorkingData** directory.
3. Restore the *Service_Activator_Home***/Config/cman.cfg** file.
4. Restore the *Service_Activator_Home***/Config/omniorb.cfg** file.

For each host where the Proxy Agent/Device Drivers are installed:

1. Ensure that you have the component names used on the previous installation to use, otherwise any device associations are lost. Consult the backed up **cman.cfg** file.
2. Reinstall the Common components, the Component Manager, the Proxy Agent, and the relevant Device Drivers.
3. Restore the **/opt/OracleCommunications/ServiceActivator/WorkingData** directory.
4. Restore the *Service_Activator_Home***/Config/cman.cfg** file.
5. Restore the *Service_Activator_Home***/Config/omniorb.cfg** file.

For each host where the OIM is installed (optional):

1. Reinstall the Common Components, Component Manager, and the OSS Integration Manager.
2. Restore the *Service_Activator_Home***/Config/omniorb.cfg** file.

For the host where the Event Handler is installed (optional):

1. Reinstall the Common Components, Component Manager, and Event Handler.
2. As the root user, restore the **/opt/OracleCommunications/ServiceActivator/WorkingData** directory.

3. If the **event_handler.props** and **event_handler.rules** files were backed up, restore these files to the *Service_Activator_Home***/Config** directory.
4. If the Event Handler is installed on a dedicated host machine, restore the *Service_Activator_Home***/Config/omniorb.cfg** file.

> **Note:** You do not need to perform the final step if the Event Handler is installed on the same host machine as the OIM.

6

Monitoring and Managing IP Service Activator

This chapter describes how to view and interpret the status of transactions, and the error messages and log information produced by Oracle Communications IP Service Activator.

Setting the Component Manager Startup Delay Time

The component manager startup delay time specifies the period, in seconds, after which the component manager attempts to start IP Service Activator components. A delay time is only applied if the component manager is run as a service.

The default delay time is 0 seconds. You should set a higher value if the Policy Server and the Oracle server are installed on the same host machine. The delay must allow sufficient time for the Oracle server to initialize on a system reboot before the component manager attempts to start IP Service Activator components.

To set the component manager startup delay time:

1. Using a text editor such as vi, open the **cman.cfg** file.

   The default location of the file is **/opt/OracleCommunications/ServiceActivator/Config**

2. Locate the **StartupDelay** setting and edit with the new value. For example, to set the startup delay time to 30 seconds, enter:

   ```
   StartupDelay 30
   ```

Setting the Component Manager Shutdown Timeout Period

The component manager shutdown timeout period specifies the time, in seconds, after an **ipsacm stop** command is executed from the command prompt, and before the component manager starts to shut down any IP Service Activator processes that are still running. The default value is 60 seconds (1 minute).

If you want to change this value, we recommend you experiment with timeout values to find one that allows the Policy Server enough time to return to a non-propagating state before shutdown.

The timeout period should be set to a value that allows the Policy Server to shut down in a non-propagating state. Setting too low a value may result in loss of data from the last propagate. The optimum timeout value is dependent on such factors as the size of the database and the number of outstanding calls being made.

Set the component manager shutdown timeout period using the Configuration GUI tool.

IP Service Activator Process Names on Solaris/Linux

You can list IP Service Activator processes by running the **ipsaps** script, which is installed by default in the **/opt/OracleCommunications/ServiceActivator/bin** directory.

The Naming Service

If you chose not to run the naming service as a service that is started automatically on reboot, you must start and stop it manually.

> **Note:** Only the IP Service Activator administrator can start and stop the naming service.

To start and stop the naming service from the shell:

1. Log on as the IP Service Activator administrator.
2. Type one of the commands in Table 6–1:

Table 6–1 Using Commands to Start and Stop the Naming Service

Command	Function
`$ ipsans start`	Starts the naming service in the background, logging information is recorded in **/opt/OracleCommunications/ServiceActivator/logs/ipsans.log**
`$ ipsans stop`	Stops the naming service

To check whether the naming service is running:

1. Run the **ipsaps** script:

   ```
   $ ipsaps
   ```

For more information about the **ipsaps** script, see "Running Troubleshooting Scripts".

The Component Manager

If you chose not to run the component manager as a service that is started automatically on reboot, you must start and stop it manually.

To start and stop the component manager from the shell:

> **Note:** Only the IP Service Activator administrator can start and stop the component manager.
>
> The naming service must be running before you can start the component manager.

1. Log on as the IP Service Activator administrator.
2. Type one of the commands listed in Table 6–2:

Table 6–2 Using Commands to Start and Stop the Component Manager

Command	Function
`$ ipsacm start`	Starts the component manager in the background, logging information is recorded in **/opt/OracleCommunications/ServiceActivator/logs/ipsacm.log**
`$ ipsacm stop`	Stops the component manager

To check whether the component manager and other components are running:

- As the IP Service Activator administrator, type:

  ```
  $ ipsaps
  ```

  The script returns a list of the IP Service Activator component processes that are currently running.

The User Interface

Note: The naming service and component manager must be running before you can start the user interface.

The user interface is run on Windows workstations. Click the IP Service Activator icon, and select the start user interface option.

Managing Validation on Policy Server Initialization

On startup, the Policy Server performs many integrity validation checks as it loads the object model from the database. No errors occur unless IP Service Activator has been upgraded, or there is a database problem.

A significant period of time is spent checking the validity of parent-child object linkages. When appropriate, you can disable the linkage validation check, and thereby significantly reduce the Policy Server initialization time.

If linkage validation faults are found, a Critical error message (2400 - Linkage errors detected during object model initialization) is reported after the Policy Server starts up.

Setting the Linkage Validation Value

Use one of the following two Policy Server command line arguments, typically specified in the **cman.cfg** file, to change the Linkage Validation setting:

Syntax:

```
-linkageValidation
+linkageValidation
```

If neither argument is set, the linkage validation check is **Automatic**.

The second argument sets the linkage validation check to **Enabled**.

If both arguments are specified, the linkage validation check is **Enabled**.

The first argument sets the linkage validation check to **Disabled**.

Values:

- **Automatic (default)**: Performs the linkage validation check only if the IP Service Activator version in the database is different than the running version of IP Service Activator. The version includes the release and build number. The linkage validation check is disabled if the versions are the same.
- **Enabled**: Always performs the linkage validation check on Policy Server startup.
- **Disabled**: Never performs the linkage validation check. Error message 2400 will not be generated for any errors.

With the Automatic setting, if a linkage error is found, the Policy Server does not update the IP Service Activator version in the database. This effectively enables the linkage validation check - it forces the Policy Server to re-run the linkage validation check on every Policy Server restart until you fix any errors. You can identify the errors by running the Database Integrity Checker and working with Oracle GCS to solve the errors. For information about running the Database Integrity Checker, refer to "Checking the Integrity of a Database".

Displaying the Linkage Validation Setting

Logs indicate whether the Linkage Validation Check is enabled, and whether the IP Service Activator version check is enabled if the Linkage Validation mode is Auto.

You can view the linkage validation setting using ComponentParameters:

```
% ComponentParameters -ComponentLocation [hostname] -ComponentName master_server
-get ObjectModel.Linkage.Validation
```

The result is as follows:

```
master_server.server@hostname[version]:
$ObjectModel.LinkageValidation=[enabled|disabled|automatic]
```

For example:

```
master_server@system4187[5.2.2.84]: $ObjectModel.LinkageValidation=automatic
```

Enabling the Linkage Validation at Upgrade Time

In general, if either the object model or the database has been upgraded, the linkage validation setting should be **Automatic** or **Enabled** to ensure that the object model is validated as correct for subsequent Policy Server restarts.

An upgrade increments the IP Service Activator version (release and/or build number). For this reason, the Database Integrity Checker (which always performs the Linkage Validation Check) is run before and after an upgrade activity.

> **Note:** The version of the Database Integrity Checker (which is given in the first line of its output summary report) must always match the installed version of IP Service Activator.

Managing components

This section describes how to discover newly-installed system components and start up and shut down components.

Discovering New System Components

The process of checking for new system components is run automatically whenever the policy server is started up. However, if any new system components are added, such as new host systems, you can run the **Find System Components** command. This queries the naming service to find any new components and update the object model.

To discover and set up new system components:

- From the **Tools** menu, select **Find System Components**.

New components are located and entered in the object model. You can view them on the **System** tab in the hierarchy pane.

Setting the Proxy Agent Timeout Value

The default time-out value for the Proxy Agent is 10800 seconds (3 hours). It is a command line parameter and is passed while starting the proxy agent. On Windows you must set it in Registry, and on Solaris/Linux you must set it in **cman.cfg** file.

To specify a different time-out value, use the following command line parameter for the Proxy Agent:

```
-gateTimeout <seconds>
```

To Restart a Halted Component Manager

You can only restart stopped components from the user interface if the component manager is still running, that is, you used the **Shutdown** option to shutdown components.

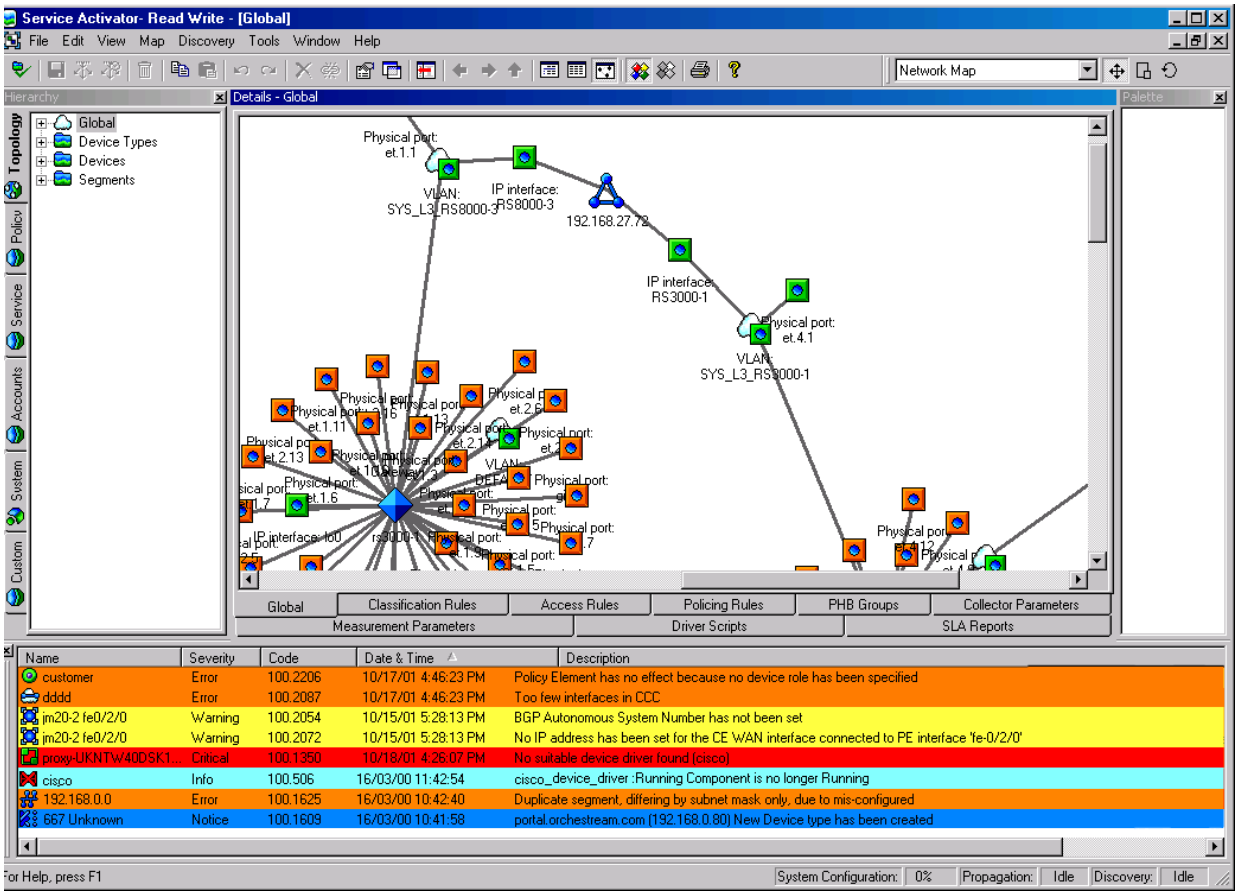
1. In the **System Hosts** folder on the **System** tab, select the appropriate system host and display its property pages.
2. Click **Startup** and click **OK**.
3. Click the **Save** button on the toolbar to implement the startup.

Viewing and Understanding Error Messages

When you validate configuration or commit a transaction, IP Service Activator reports any errors in the configuration in the current faults pane. A message may be one of the following types: critical, error, warning, notice, or information. Each message type is color coded for quick reference.

Note: Best practice: Check the current faults pane, as shown in Figure 6–1, after committing each transaction and correct errors that are reported.

Figure 6–1 IP Service Activator GUI: Current Faults Pane



The pane displays information, warning, and error messages. Most of these messages result from network discovery or data validation.

Table 6–3 describes how the background color of a message indicates its type:

Table 6–3 Recognizing Error Message Types

Color	Description
Red	Critical faults requiring intervention: failure of an IP Service Activator component or a serious problem with a device.
Orange	Errors: serious faults in the policy system, but not in a system component.
Yellow	Warnings, such as validation errors.
Blue	Notices: some user action is required.
Cyan	Information only: no user action required.

For each message, IP Service Activator displays information in the columns shown in Table 6–4:

Table 6–4 Displaying Error Message Information

Column	Description
Name	Name of the object in which the fault occurred. The icon of the object also appears.

Table 6–4 (Cont.) Displaying Error Message Information

Column	Description
Severity	Critical, Error, Warning, Notice or Information, as described above.
Code	Code number of the message. Numbers prefixed with 100 indicate those generated from IP Service Activator components.
Date & Time	Date and time that the error occurred.
Description	Text of the message.

Getting More Information About Error Messages

IP Service Activator online Help provides information about each of the IP Service Activator error messages. You can obtain help on any error message listed in the current faults pane by selecting the error and pressing F1. For each message, the help topic describes its type, severity level, description, and suggested action to remove the error.

Suppressing Error Messages

You can suppress object model warning messages by editing the object model messages xml file. By default, this file is *Service_Activator_Install***Config\messages_default.xml**

Set **suppress="true"** for each warning that should no longer be raised. For example:

```
<ErrorMessage id="2073" code="IDS_NO_STATIC_ROUTES_CONFIGURED" text="Static
routing has been chosen for VPNs, but no static routes have been set up"
severity="Warning" suppress="true"/>
```

You need to modify this file on each host where the graphical user interface (GUI) is installed, and on the host where the policy server is installed. To implement the change, restart the policy server and each affected GUI instance.

Logging into IP Service Activator

IP Service Activator produces a number of log files that you can use to find out what is happening within the system and to help troubleshoot problems.

Setting Up Logger Messages on Solaris/Linux

Some pre-configuration of Solaris/Linux OS settings is required to manage system logger messages generated by IP Service Activator. See "Additional Setup Tasks to Be Run Prior to Initial Startup" for details.

Checking the System Logs

IP Service Activator produces a number of system logs:

- System Message Log: lists all errors, warnings, and information messages reported by the system.

> **Note:** The System Message Log uses a timestamp based on local time. Other logs use UTC for their timestamps. In order to be able to view System Logs, ODBC must be configured on the Windows GUI machine. The ODBC settings must match the database settings in the cman.cfg file on the Policy Server machine.

- Device Configuration Log: lists all changes to policy implemented at the devices.
- Configuration GUI: Configuration GUI tool client and server logs
- Audit Trail: lists changes made to the object model.

Logging is performed by the system logger component, which resides with the policy server. You can control which logs are maintained and how long logged details are kept before being discarded. You can set logging details as a property of the policy server or for each individual log type.

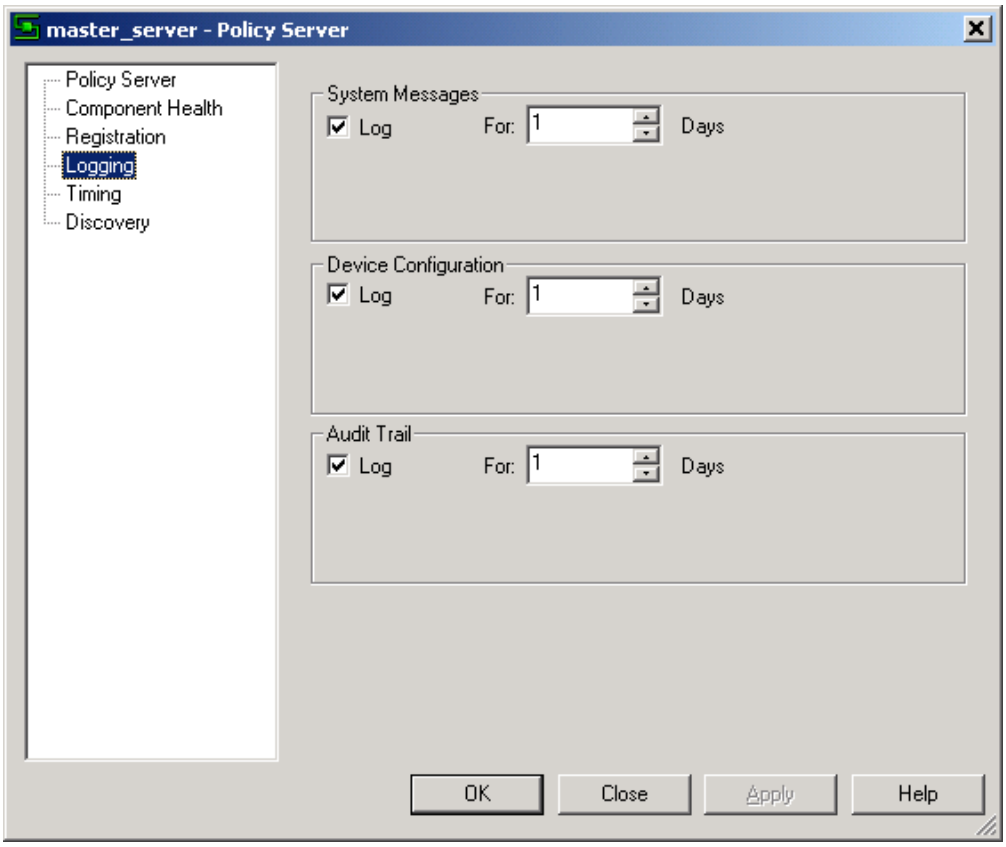
> **Note:** The System Messages and Device Configuration logs are turned on by default, with a setting of 1 day. The Audit Trail is turned off by default, since maintaining a full audit trail can have an effect on system performance.
>
> It is also possible to enable debug log output for components. For more information, see "Generating debug log files for non-Java based IP Service Activator components".

To access the system logs:

1. On the **System** tab, open the **System Logs** folder.
2. Double-click one of the listed logs to display its content in the details pane.

To set up or change the settings for all system logs:

1. On the **System** tab, select the **Master Server** object.
2. Select **Properties** from the pop-up menu.
3. Select the **Logging** property page.



4. For each log (**System Messages, Device Configuration,** and **Audit Trail)**, select the **Log** check box to specify that relevant events are logged. Specify the number of days (in the range 1 to 28) for items to remain in the log before being discarded.
5. Click **OK**.

To set up or change the settings for a specific system log:

1. On the **System** tab under the **System Logs** folder, select the object whose log details you want to set.
2. Select **Properties** from the pop-up menu.
3. Select the **Log** check box to specify that relevant events are logged. Specify the number of days (in the range 1 to 28) for items to remain in the log before being discarded.
4. Click **OK**.

System Message Log

The System Message log holds system messages: this includes all errors, warnings, and information messages, apart from data validation messages.

> **Note:** Current errors and faults, including validation messages, are also reported on the current faults pane. See "Viewing and Understanding Error Messages".

To view the System Message log:

- In the **System** tab, **System Logs** folder, double-click **System Message Log**.

  The log is displayed in the details pane. Messages appear in the order in which they were raised, but you can sort by a selected column by clicking on the appropriate heading.

  To refresh the log display, press F5 or select **Refresh** from the pop-up menu of the log.

For each message, the IP Service Activator system displays the types of information described in Table 6–5:

Table 6–5 The System Message Log Display

Heading	Description
Time	The date and time that the message was logged.
Severity	The severity level of the problem. One of the following: **Critical:** Indicates a serious fault in a system component or a device, requiring user intervention. **Error:** Indicates a serious fault in provisioning policy or services. **Warning:** A less serious fault. **Notice:** Indicates that some user action is required. **Info:** Informative message only; no action required.
Message Type	The type of message. One of the following: **Fault:** A component has generated a fault. Most faults require some user action in order to fix them. **Clear:** (specific component or Clear All) An existing fault has been cleared, either by the system or by manual intervention. **Message:** Reports a system event that generally does not require user action.
Code	Message code number. Numbers prefixed with 100 indicate those generated from IP Service Activator components. Other prefixes indicate messages originating from components that have not been supplied by IP Service Activator, for example, cartridges produced by a third party. For more information about these messages, contact your software supplier.
Description	The text of the message.

Messages are color-coded.

- Faults are color-coded according to severity:
  - Critical faults appear on a red background.
  - Errors appear on an orange background.
  - Warnings appear on a yellow background.
  - Notices appear on a blue background.
  - Info messages appear on a cyan background
- Clears appear on a green background.
- Messages appear on a white background.

To display more information about a particular message:

- Select it and click **F1** or select **Help** from the pop-up menu.

To update the System Messages window to display the latest messages:

- Right-click anywhere in the window and select **Refresh** from the pop-up menu or press F5 or select **Refresh** from the pop-up menu of the log.

Device Configuration Log

The Device Configuration Log records all changes made to policy and services; that is, details of the locations and times at which tasks occur, for example:

- A concrete rule is installed, uninstalled, or fails
- A concrete PHB group is installed, uninstalled, or fails
- A concrete VPN is installed, uninstalled, or fails
- A concrete CCC is installed, uninstalled, or fails
- NetFlow or SAA configuration is installed, uninstalled, or fails

To view the Device Configuration Log:

- In the **System** tab, **System Logs** folder, double-click **Device Configuration Log**.

  All logged policy changes are listed in the details pane.

For each configuration change, the information in Table 6–6 is displayed:

Table 6–6 The Device Configuration Log Display

Heading	Description
Date & Time	Date and time of the configuration.
Event Type	One of the following: **Installed:** The configuration has been propagated to proxy agents and successfully installed on the device. **Uninstalled:** The configuration group has been uninstalled. This can mean that the rule is not currently active (for example, because the date and time or state variable limitations do not apply at present) or the rule has been deleted or disabled by a user. **Failed:** The proxy agent experienced a failure trying to install the configuration on the device, and it has been discarded.
Device Name	Name of the device to which configuration has been applied.
Interface	Name of the interface to which configuration has been applied.
Description	Type and name of the IP Service Activator object associated with the configuration.
Identity	Internal ID number of the object.

To update the Device Configuration log window to display the latest messages:

- Right-click anywhere in the window and select **Refresh** from the pop-up menu or press F5 or select **Refresh** from the pop-up menu of the log.

Checking the Network Processor Logs

Each installed network processor records in a specific log file all device configuration changes that it makes. You can check these log files to see the configuration changes made when policies and services are propagated. On Solaris/Linux systems, the log files are created in **/opt/OracleCommunications/ServiceActivator/AuditTrails**.

Each file is named as follows:

*<np><cartridge>***.audit.log**

For example, **npCisco.audit.log** is the log file for the Cisco IOS network processor cartridge.

Log files are text files, recording the date, time, and details of each configuration change made to the devices controlled by the network processor. Each time a network processor starts, a welcome message is written to the log file. For example:

```
2015-06-09 14:23:44|#AuditLogging: Opened
```

Each configuration entry is preceded by the date and time of the change and the IP address of the device to which the configuration was sent. Times are expressed in Coordinated Universal Time (UTC).

```
Thu Apr 13 11:24:07 2000|192.168.0.203| <config details>
```

The text logged for configuration changes varies according to the individual network processor cartridges, because the cartridges use different mechanisms to communicate with the devices.

Configuration GUI Log Files

The log files for the Configuration GUI are created on the host machine, and are located in the following directories:

- On Solaris/Linux:

  /opt/OracleCommunications/ServiceActivator/logs

- On Windows:

  C:\Program Files\OracleCommunications\ServiceActivator\logs

The log files are **configToolServer** (for server) and **configToolClient** (for the GUI client).

Based on the **LOG_LEVEL** parameter in the startup script the logs are captured. The **LOG_LEVEL** parameter can have any one of the following values:

- SEVERE
- WARNING
- INFO
- DEBUG
- TRACE
- TRACE1
- TRACE2

The default level is INFO.

Managing the Size of System Log Files

Use the following recommendations to manage the size of the system log files.

The **cisco.log**,**cman.log**, **cman.log**, **event_handler.log**, **integration_manager.log**, **ipsacm.log**, **ipsans.log**, **juniper.log**, **networkprocessor.log**, **policy_server.log**,

proxy.log, **system_log.log**, **unisphere.logusage.log** and **cman_syslog_error.log** files do not grow substantially as you use IP Service Activator.

To manage the size of logs such as **policy_server.log**, **integration_manager.log**, **system_log.log**, **proxy.log** and the **device driver logs**, add the following entry to the **cman.cfg** file:

-debugFileMaxSize <*size in bytes*>

Viewing Log Files

The log information is intended to be viewed with an external Log4J compliant viewer such as Chainsaw. You can configure the viewer to view log entries directly out of the IP Service Activator database. Refer to the vendor's product documentation to configure the viewer.

Network Processor Logging

This section explains how logging is implemented and configured on the Network Processor.

Network Processor Log Files

The Network Processor records log files in **/opt/OracleCommunications/ServiceActivator/logs** and audit logs in the **/opt/OracleCommunications/ServiceActivator/AuditTrails** directory. There is just one current log file and one current audit file (per cartridge) for each Network Processor. Network Processor log and audit files roll over when they exceed a configurable maximum size.

The log and audit file format is also different, because the Network Processor logs use the **log4j** logging utility. This utility allows flexible configuration of logging levels and other logging attributes.

Setting Up Logging

Pre-requisite: The Network Processor is installed and verified.

The Network Processor logger is based on **log4j**, a Java utility with rich logging capabilities. Using log4j, you can perform logging level filtering, with file rollover based on file size. Procedures for configuring these items are provided later in this section.

The **logging.properties** file contains the log4j configuration parameters. For details see "Logging Properties".

Note: For log4j customization information (including formatting the log output, redirecting the log output to stdout or a rolling file), refer to the log4j online documentation:
`http://logging.apache.org/log4j/2.x/project-info.html`

Checking Network Processor Logs

Network processor logs are written to:

/opt/OracleCommunications/ServiceActivator/logs/networkprocessor.log

The default level of logging is at the **INFO** level, a medium logging level (see " Logging Levels").

Additional log viewer applications, such as Chainsaw version 2.0, may provide more readable output of log records.

Checking the Network Processor Cartridge Log

The audit trail logs for network processor cartridges record the commands sent to devices. Audit trail logs are written to:

/opt/OracleCommunications/ServiceActivator/AuditTrails/np*cartridge_Name***.audit.log**

The following is an example of part of audit log file:

```
2009-03-24 04:39:37|#AuditLogging: Opened
2009-03-24 04:39:37|10.156.68.12|#Start Configuration
2009-03-24 04:39:37|10.156.68.12|file-interface|#Applying Configuratio
2009-03-24 04:39:38|10.156.68.12|file-interface|terminal length 0
2009-03-24 04:39:38|10.156.68.12|file-interface|conf t
2009-03-24 04:39:38|10.156.68.12|file-interface|vc-class atm VcClass_
2009-03-24 04:39:38|10.156.68.12|file-interface|ubr 22500
2009-03-24 04:39:38|10.156.68.12|file-interface|exit
2009-03-24 04:39:38|10.156.68.12|file-interface|interface ATM2/0
2009-03-24 04:39:38|10.156.68.12|file-interface|class-int VcClass_0
2009-03-24 04:39:38|10.156.68.12|file-interface|exit
2009-03-24 04:39:38|10.156.68.12|file-interface|alias exec IpsaConfigVersion
2009-03-24T08:39:36.301Z
2009-03-24 04:39:38|10.156.68.12|file-interface|end
2009-03-24 04:39:38|10.156.68.12|file-interface|copy running-config startup-config
2009-03-24 04:39:38|10.156.68.12|file-interface|logout
2009-03-24 04:39:38|10.156.68.12|#End Configuration
```

Format Syntax

[yyyy-mm-dd
hh:mm:ss | *<mgmt-ip-address>* | *[delivery-mode-tag* | *](command* | *#<status-msg>*

where,

- *yyyy-mm-dd hh:mm:ss:* Date and time of the change. Time is expressed in Co-ordinated Universal Time (UTC).
- *mgmt-ip-address*: IP address of the device to which the configuration was sent
- *delivery-mode-tag*: Possible values are:
  - file-interface or offline-maintenance
  - no-command-delivery or offline-test
  - enforce (shown when Command Re-issue feature used)

  For more information about delivery mode tags, refer to the *Online Help* and the "Command Delivery Modes". For more information about the Command Re-issue feature, refer to the *Online Help*.
- *command*: For some cartridges, commands may actually be multi-line XML fragments or documents.
- *status-msg:* Possible values are:

- #AuditLogging: Opened. Cartridge is opened and recognized by the network processor and is ready to log commands.
- #Start Configuration. A configuration session is started with the device.
- #Applying Configuration. A configuration is being applied to a device.
- #Start Enforce. A modification to the device configuration is being started after a command re-issue request
- #End Enforce. The modification to the device configuration is finished.
- #End Configuration. The configuration session with the device is ended.
- #Rollback Configuration. A command delivery error or an error response from the router to a delivered command is encountered. A rollback procedure is initiated, the failing service is quarantined, and an attempt is made to deliver any remaining configuration changes.

Logging Levels

Log levels in log4j included items logged at all higher levels. For example, with logging set to the WARN level, items logged include those logged at the ERROR and FATAL levels.

Table 6–7 summarizes usage for each logging level:

Table 6–7 Logging Levels

Level	Usage
FATAL	Very severe errors that will presumably lead the application to crash. For example: Unable to initialize ORB.
ERROR	Error events that may be recoverable or allow the application to continue. For example: Provisioning of device fails, as well as the rollback. The device is in an 'unrecoverable' state, but the application itself is able to continue and should not interrupt configuration of other devices.
WARN	Indicates potentially harmful conditions. For example: Received a warning back from the device during configuration.
INFO (default)	Highlight the progress of the application at a coarse-grained level. For example: Committing a device/transaction.
DEBUG	Fine-grained events only useful for debugging the application. For example: Indicating the specific steps during the commit: creation of each model, annotation, sending commands to device.
TRACE1	Additional information only useful when performing specific debugging tasks which may require more extensive output. For example: Logging of the various models when they are created
TRACE2	Only for very fine-grained logging, not normally used since it would flood the logs. For example: Outputting the models during the various stages of their construction.

Changing the Logging Level Filter

To change the Network Processor logging level, or the audit trail logging level, edit the **logging.properties** file, located in directory **/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor/.**

For example, to change the logging level from **Info** to **Debug**, modify the line:

```
log4j.rootLogger=info, file
```

to:

```
log4j.rootLogger=debug, file
```

Similarly, to change the audit trail logging level from **Info** to **Debug**, modify the line:

```
log4j.logger.com.metasolv.serviceactivator.networkprocessor.AuditLogger=info,
audit
to: log4j.logger.com.metasolv.serviceactivator.networkprocessor.AuditLogger=debug,
audit
```

Managing Log File Rollover

The log files are configured to roll over to a new log file when they reach the maximum file size (MaxFileSize), which you can adjust by editing the **logging.properties** file. The default network processor log file size is 50 MBytes. The default audit log file size is 8 MBytes.

When maximum file size is reached, the log file is closed, and a new log file is opened to record logs. Each closed log file includes an integer value in its filename. This integer value is incremented for each newly closed log file until the value of MaxBackupIndex is reached. Then the logfilename integer resets to 1, and the previous log file of the same name is over-written. You can adjust the number of previous log files to keep on disk. The default number of previous versions is 1.

It may be appropriate to balance these two parameters to meet your logging requirements. For example, you can choose to store many small log files, or a few large log files.

To adjust log file size and number of previous versions:

1. To change the Network Processor log file size, or the audit log file size, modify the value of **MaxFileSize** in the appropriate line of the **logging.properties** file:

   ```
   log4j.appender.file.MaxFileSize=8MB
   log4j.appender.audit.MaxFileSize=8MB
   ```

2. To change the number of previous versions of the Network Processor log file or the audit log file, modify the value assigned to **MaxBackupIndex** in the appropriate line of the **logging.properties** file:

   ```
   log4j.appender.file.MaxBackupIndex=1
   log4j.appender.audit.MaxBackupIndex=1
   ```

Using the dailyFileAppender

> **Note:** Alternatively, you might choose to use the dailyFileAppender for network processor or audit logs. It rolls log files over on a daily basis, so each log file could potentially be large in size. Because log files are not deleted, you might use a lot of storage for the log files if you do not delete them manually.

For details, refer to the log4j online documentation:
http://logging.apache.org/log4j/2.x/project-info.html

Log Output

The log output in text form contains several fields that are defined in the logging.properties file. See "Logging Properties" for examples of the field definitions.

The following is a sample output line in the log file and an explanation of the contained fields.

```
20061206-094547|575|WARN|Thread-0|10.13.4.144|cisco|1002|proxy_
np|hostname-1|Pre-Check for Static Route configuration failed: next hop ip 1.1.1.1
is not routable.
```

Explanation of Fields:

20061206-094547: timestamp of when the log was raised.

575: millisecond value of the log timestamp.

WARN: logging level for the specific log.

Thread-0: name of the thread that generated the log.

10.13.4.144: IP address of device to which log applies

cisco: name of the cartridge

1002: Association ID

proxy_np: Component generating log

hostname-1: Server generating log

Logging Properties

The log4j logging configuration is set up in the **logging.properties** file:

/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor/logging.properties

> **Note:** Do not modify properties in the **Internal Use** section of the **logging.properties** file.

Refer to the log4j online documentation for details on specific log4j configuration attributes:

http://logging.apache.org/log4j/2.x/project-info.html

LogReader

The LogReader collates log entries from multiple IP Service Activator components and instantly uploads them into the Oracle database. You must install the LogReader on each IP Service Activator machine where you have at least one component running and whose logs you want to consolidate into the database. The LogReader expects all IP Service Activator component logs to be in the default format - customized formats are not supported.

Configuring the LogReader Component

The LogReader component can be configured using the following property file:

/opt/OracleCommunications/ServiceActivator/Config/java/com/metasolv/serviceactivator/logreader/logCollector.properties

The following are guidelines used to configure the LogReader component:

- Oracle Database is the only database supported. JDBC is the only connection mechanism supported.
- Choose either the Oracle Database Standard Edition or Enterprise Edition. The default assumption is that you have the Standard edition.
- **fileRollCheckInterval**: This parameter defines how often each input log file should be checked to see if it has rolled over (milliseconds). The default is one minute.
- **listenSocketPort**: This parameter defines where the LogReader should listen for log4j events. The default is 4446. This option should be used with caution since socket-based logging can have a significant performance impact on the logged component.
- **logLifetime**: This parameter defines the number of days logs are kept. The logs for each day is kept in its own partition. The default is seven days.
- **maintenanceInterval**: This parameter defines how often the LogReader should perform database maintenance. This process deletes partitions older than the defined logLifetime parameter. Values can be specified in milliseconds. The default is one hour.

Configuring the Scope of the LogReader

The components for which logs will be collated by the LogReader can be configured using the following property file:

/opt/OracleCommunications/ServiceActivator/Config/java/com/metasolv/serviceactivator/logreader/default.properties

This file is automatically populated during IP Service Activator installation and contains information on all standard components installed on that host. This property file can be modified anytime after the installation. Modifying the property file requires a restart of the LogReader to re-read the property file, which has the following format:

```
components = <comma separated list of components that are to be monitored:
name1,name2>
component.<name1>.name = <component-name>
component.<name1>.filename = <absolute-path-to-component's-log-file>
component.<name1>.newLogFileFormat = <true|false>

component.<name2>.name = <another-component-name>
component.<name2>.filename = <absolute-path-to-second-component's-log-file>
component.<name2>.newLogFileFormat = <true|false>
Example:
components = cman,cman_syslog_error,ipsacm,network_processor
component.cman.name = Component Manager
component.cman.filename = <ServiceActivatorHome>/logs/cman.log component.cman_
syslog_error.name = Component Manager Syslog Error component.cman_syslog_
error.filename = <ServiceActivatorHome>/logs/cman_syslog_error.log
component.ipsacm.name = ipsacm
<INSTALLATION_DIRECTORY>/ServiceActivator/logs/ipsacm.log component.network_
processor.name = Network Processor component.network_processor.filename =
<ServiceActivatorHome>/logs/networkprocessor.log
component.network_processor.newLogFileFormat = true
```

The **newLogFileFormat** tells the LogReader whether the log file being read is the legacy C++ IP Service Activator format or the new Java IP Service Activator format. If

this line is missing, it is assumed to be in the legacy C++ IP Service Activator format. Therefore, you have to add this line only for Java components and set it to true.

Database Schema for LogReader

The records for the tables are stored in daily partitions based on the insert_date field. The partitions are dropped according to a sliding window specified by the logLifetime property in the: **/opt/OracleCommunications/ServiceActivator/Config/java/com/metasolv/serviceactivator/logreader/logCollector.properties** file.

> **Note:** If you choose the database server to be Oracle Enterprise Edition, the partitions are created. If you choose Standard Edition, the records are dropped individually since partitions are not supported.

The enhanced logging facility for the Network Processor generates tables in the Oracle Database with the schema shows in Table 6–8.

Table 6–8 Using the Database Schema for LogReader

Table	Column	Notes
logging_event	insert_date	date not null
logging_event	sequence_number	number(20) not null
logging_event	timestamp	number(20) not null
logging_event	rendered_message	varchar2(4000) not null
logging_event	logger_name	varchar2(254) not null
logging_event	level_string	varchar2(254) not null
logging_event	ndc	Nested Diagnostic Context, varchar2(4000)
logging_event	thread_name	varchar2(254)
logging_event	reference_flag	Smallint
logging_event	caller_filename	varchar2(254) not null
logging_event	caller_class	varchar2(254) not null
logging_event	caller_method	varchar2(254) not null
logging_event	caller_line	char(4) not null
logging_event	event_id	number(10)
logging_event_attribute	insert_date	date not null
logging_event-attribute	event_id	number(10) not null
logging_event-attribute	device_ipa	varchar2(15); default null
logging_event-attribute	cartridge_name	varchar2(200); default null
logging_event-attribute	component_name	varchar2(100) not null
logging_event-attribute	host_name	varchar2(100) default null
logging_event-attribute	insert_date	date not null
logging_event-attribute	event_id	number(10) not null
logging_event-attribute	assoc_id	number(10) not null

Table 6–8 (Cont.) Using the Database Schema for LogReader

Table	Column	Notes
logging_event_exception	insert_date	date not null
logging_event_exception	event_id	number(10) not null
logging_event_exception	i (this is the stack trace index)	smallint not null
logging_event_exception	trace_line	varchar2(254) not null

Using the view **v_logging_event**, you can see all the tables listed above sorted together by the **event_id**.

All the elements in the table above (with the exception of the entire logging_event_ attribute table) come from the log4j standard. More information is available at http://logging.apache.org/log4j/.

The l**ogging_event_attribute** table stores non-standard IP Service Activator attributes:

- **device_ipa**: This is the management IP address of the device being operated in the associated log event, if the event can be traced to a particular device. This is applicable only to logs originating from the Network Processor.
- **cartridge_name**: If the log event originates from a particular cartridge or from the framework executing on behalf of a particular cartridge, the cartridge name (as specified in the cartridge registry) will be captured here.
- **component_name**: The name of the IP Service Activator component that generated the log event is stored here.
- **host_name**: The host name of the originating LogReader component is stored here. Because the logs of each component are handled by a co-located LogReader, this is also the hostname of the component that generated this log event.

Records in this table are tied to those in other tables via **event_id**.

Network Processor Database-related Log Files

On the IP Service Activator server, the following files in the directory *Service_ Activator_Home*/**logs** contain entries related to the Network Processor-database software:

```
networkprocessor.log
logreader.log
upgradetool.log
```

The logging level for these 4 files is determined by an entry in the following files respectively:

Service_Activator_ Home/**Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor/logging.properties**

Service_Activator_ Home/**Config/java/com/metasolv/serviceactivator/logreader/logging.properties**

Service_Activator_ Home/**Config/java/com/metasolv/serviceactivator/logwriter/logging.properties**

Service_Activator_Home/**Config/networkProcessor/upgradeTool/logging.properties**

On the Oracle database server where the Network Processor schema is hosted, the **ipsa_plsql_dir** (example: **/opt/OracleCommunications/ServiceActivator/logs**) contains daily log files with debug entries generated by the PL/SQL stored procedures in the Network Processor package. These files are named as follows:

*YYYYMMDD*_np.log (example: 200700518_np.log)

Log File Cleanup: Routine Maintenance Task

Periodically, you should clean up these log files. Their size is small, especially if debugging is turned off, so they are unlikely to fill a disk, but you should monitor their size and perform a cleanup as part of your routine maintenance schedule.

Setting the Debug Log Level

The logging level for these PL/SQL logs is stored in the np_debug table as a binary value: **0=off** (default on schema creation) or **1=on**. When it is turned off, log entries will only be created if an error is encountered.

To determine the current value for the debug log level, enter:

```
npAdmin.sh get_plsql_debug
```

To change the debug log level, enter:

```
npAdmin.sh set_plsql_debug on|off
```

Setting the Archive Limit for Transactions

You can view the list of committed transactions on the **System** tab under the **Transactions** folder and specify the number of committed transactions that IP Service Activator maintains in its transactions archive.

If you specify an archive limit, IP Service Activator also applies an upper threshold of 25 for committed transactions over and above the specified limit. The total number of archived transactions is therefore the combined value of the defined archive limit and the hard-coded upper threshold. For example, if you specify an archive limit of 100, IP Service Activator deletes committed transactions only when their number exceeds 125.

> **Note:** When you set the archive limit, if more than 25 committed transactions are in the archive, none of these transactions will be deleted. This is due to the IP Service Activator application of a hard-coded upper threshold of 25 above the specified archive limit.

By default, IP Service Activator maintains information about the last 200 committed transactions. If you lower the limit, any committed transactions above the limit (plus the hard-coded threshold) are permanently deleted from IP Service Activator.

To set the archive limit for transactions:

1. From the **Tools** menu, select **Options**.

   The **Options** dialog box opens.

2. Select the **Transactions** property page.

3. In the **Transaction archive limit** field, specify a new archive limit.

   The default is 200.

Audit Trail

The Audit Trail records all changes made to the system database, that is, creating, deleting, amending or linking objects.

To view the Audit Trail:

- In the **System** tab, **System Logs** folder, double-click **Audit Trail**.

  The log is displayed in the details pane.

For each entry, the information described in Table 6–9 is displayed:

Table 6–9 The Details Pane Display

Heading	Description
Date & Time	Date and time that the user action occurred.
Username	For user operations, the name of the logged-on user who carried out the operation. For system operations, the name of the policy server.
Op Type	The type of operation performed: Begin Transaction, Modify, Create, Delete, Link, or Unlink.
Parent Type	The object affected. For Create, Modify, and Delete operations, this is the object being directly affected. For Link and Unlink, this is the parent object.
Parent ID	Internal ID number of the Parent Type object.
Child Type	The internal ID number of the Child object, that is, the object being linked or unlinked. This only appears for Link and Unlink operations.
Child ID	Internal ID number of the Child Type object, where relevant.
Parameters	For **Create** and **Modify** operations, lists the attributes and values of the affected object.

To update the Audit Trail window to display the latest messages:

- Right click anywhere in the window and select **Refresh** from the pop-up menu or press F5 or select **Refresh** from the log pop-up menu.

Policy Server Performance Tuning

As transactions are committed in the user interface or via the OIM, the policy server queues the operations they perform and processes each operation in turn. An operation is one of the low-level actions that feature in a transaction – for example, creating an object, or linking one object with another.

You can monitor the content of the database writer queue by:

- Specifying a queue size high water mark value: information is written to the operating system log file
- Switching on debug logging for the policy server: information is written to the policy server debug log file

Note: We generally recommend that debug logging is enabled for the policy server only for troubleshooting purposes.

By default, users can continue working in the user interface while the policy server processes transactions and saves information to the database. However, it is also possible to run the user interface in 'confirmed commit mode'. In this mode, users cannot commit a transaction until the previously committed transaction has been processed and its changes saved to the database. For more information, see "Running the User Interface in Confirmed Commit Mode".

Device Role Check Option

When the devices/configuration have the same roles, and there are a number of devices in the system, the concrete calculation with device role check is slower than the calculation without device role check. To address this issue, an option called ConcreteDeviceRoleCheck is provided in the database to control the concrete calculation. This option is enabled by default. This implies that the concrete calculation will have device role check. The option can turned off through policy server command line option **-ConcreteDeviceRoleCheck** disabled or policy server component parameter **policyServer.Concrete.DeviceRoleCheck**. This is better explained with the following example:

```
ComponentParameters -ComponentName master_server -set
PolicyServer.Concrete.DeviceRoleCheck disabled
```

Specifying a Queue Size High Water Mark

It is possible to monitor the number of operations in the database writer queue via the operating system log file. Queue size information is written to the file if the **QueueHighWaterMark** parameter is specified on the policy server command line. The parameter defines the number of operations that must be in the queue before information is written to the log file.

When the policy server database writer queue reaches the value specified by the **QueueHighWaterMark** parameter, an operating system logger message is raised and the high watermark level is increased by one.

If the database writer queue reaches the increased value, another operating system logger message is raised and the high watermark level is increased by one, and so on. This information is also stored in the Policy Server log file.

The default high watermark value is 1000.

It is also possible to set the maximum size of the database writer queue. When this maximum is reached, the policy server does not accept new operations into the queue while it processes the backlog. Reaching the maximum limit therefore results in some performance degradation.

The default maximum queue size is 2000.

To set the queue size high water mark and/or maximum queue size:

The queue size high water mark is defined by the following parameter:

```
-QueueHighWaterMark value
```

where *value* is the number of transactions that must be in the database writer queue before the operating system logger error message is raised. The default value is 1000 transactions.

The absolute maximum number of transactions to permit in the queue is defined by the following parameter:

```
-QueueSize value
```

where *value* is the maximum number of transactions permitted in the database writer queue. The default value is 2000 transactions.

- On Solaris/Linux platforms, edit the policy server entry in the configuration file **cman.cfg** in **/opt/OracleCommunications/ServiceActivator/Config**

  For example:

```
/opt/OracleCommunications/ServiceActivator/bin/policy_server
"-QueueHighWaterMark 1500 -QueueSize 3000 … "
```

Writing Queue Information to the Policy Server Log File

The following information about the database writer queue is written to the policy server when debug logging is turned on for the component:

- When an operation is added to the queue the current queue size and total number of operations ever added to the queue is logged.
- When an operation is removed from the queue (that is, the operation has been written to the database) the current queue length is logged.
- If the maximum queue length is reached (the default is 100) a message is written to the debug log file stating that the queue is full and that some degradation of server performance can be expected.

For information on enabling debug logging for a component, see "Generating debug log files for non-Java based IP Service Activator components".

About Transaction Status Monitoring

Transaction status monitoring provides feedback about the success or failure of a transaction so that you can view the status of transactions. See "Viewing Provisioning Statuses" for more information.

The Policy Server sends all the commands for each concrete in a transaction to the Network Processor. The Network Processor runs the commands and determines the success or failure for each transaction.

After the Policy Server receives notification from the Network Processor about all the concretes in the transaction, it automatically sets a provisioning status value for the entire transaction.

Transaction status monitoring monitors a transaction until all the concretes transition from a pending state. Monitored transactions are not deleted as part of the transaction archive limit. See "Setting the Archive Limit for Transactions" for more information.

The following list describes the parameters you can set to configure transaction status monitoring. See "Configuring Transaction Status Monitoring" for information about how to set the parameters.

- Timeout Interval: configures the amount of time that IP Service Activator tries to route and monitor a transaction before setting the transaction's provisioning status

to timedout. The default value is 900 seconds and the maximum value is 14300 seconds.

- Fault Wait Time: configures the amount of time that the Policy Server waits before collecting the faults for a failed transaction. The wait ensures that all relevant faults have propagated to the OSS Integration Manager (OIM). The timer starts when the status of all the concretes in a transaction have changed from pending to either succeeded or failed. The default value is 10 seconds.
- Poll Interval: configures how often transaction status monitoring checks the states of the concretes in a transaction. The default value is 60 seconds.

Viewing Provisioning Statuses

To view the transaction provisioning status:

1. Commit the transaction. See *IP Service Activator User's Guide* for details.
2. Verify that the transaction is committed:
   a. Click the **Systems** tab.
   b. Select **Committed Transactions**.

      The list of committed transactions appears.
   c. Verify that the transaction appears in the list.

      Note: If the transaction results in a null change (changing something to the value it is already set to), no transaction is generated.
3. Check the transaction provisioning status:
   a. Right-click the transaction and select **Properties**.

      The Properties window appears.

      The provisioning status of the transaction is displayed in the **Provisioning Status** field. Table 6–10 lists the possible provisioning status values of transactions and their meanings.

Table 6–10 Provisioning Status Values

Value	Meaning
Succeeded	The Network Processor successfully ran the commands for all the concretes in the transaction and sent the status of each concrete to the Policy Server. The Policy Server set the transaction provisioning status and removed the transaction from its cache.
Failed	At least one concrete in the transaction was rejected. The Network Processor notified the Policy Server of each affected concrete. The Policy Server set the transaction provisioning status and removed the transaction from its cache. For example, if you generate two transactions that modify the same concrete, the transaction fails because IP Service Activator cannot determine which transaction to update to what status, as one transaction could fail and one could succeed.
Timedout	The Policy Server did not receive notification from the Network Processor about all of the concretes in the transaction during the maximum time period set by the parameter TransactionMonitoringTimeout.

Table 6–10 (Cont.) Provisioning Status Values

Value	Meaning
Pending	The original state of the committed transaction while the Policy Server is waiting to receive notification about all the concretes in the transaction.

See "Transaction Troubleshooting" for more information about troubleshooting failed transactions.

Configuring Transaction Status Monitoring

Transaction status monitoring can be configured by setting parameters using the command line before IP Service Activator is running, or by setting component parameters after IP Service Activator is running.

Configuring Transaction Status Monitoring Using Command Line

To configure transaction status monitoring using the command line:

1. Set any relevant parameters using the commands listed in Table 6–11. See "About Transaction Status Monitoring" for more information about these parameters.

Table 6–11 Transaction Status Monitoring Command Line Parameters

Parameter Name	Command Line parameter
Timeout interval	`-TransactionMonitoringTimeout` *value_in_seconds* where *value_in_seconds* is the timeout value in seconds
Fault wait time	`-FaultWaitTime` *value_in_seconds* where *value_in_seconds* is the fault wait time value in seconds.
Poll interval	`-TransactionMonitoringPollInterval` *value_in_seconds* Where *value_in_seconds* is the poll interval value in seconds

Configuring Transaction Status Monitoring After IP Service Activator is Running

To configure transaction status monitoring after IP Service Activator is running:

1. Run the component parameter utility.
2. Set any relevant parameters using the commands listed in Table 6–12. See "About Transaction Status Monitoring" for more information about these parameters.

Table 6–12 Transaction Status Monitoring Component Parameters

Parameter Name	Component Parameter
Timeout interval	`-ComponentParameters -`*component_name* `master_server -setTransactionMonitoring.Timeout` *value_in_seconds* where: ■ *value_in_seconds* is the timeout value in seconds. ■ *component_name* is the name of the component.
Fault wait time	`-ComponentParameters -`*component_name* `master_server -setTransactionMonitoring.FaultWaitTime` *value_in_seconds* where: ■ *component_name* is the name of the component. ■ *value_in_seconds* is the fault wait time value in seconds.

Table 6–12 (Cont.) Transaction Status Monitoring Component Parameters

Parameter Name	Component Parameter
Poll interval	`-ComponentParameters -`*`component_name`* `master_server -setTransactionMonitoring.PollInterval` *`value_in_seconds`* where: ■ *component_name* is the name of the component. ■ *value_in_seconds* is the poll interval value in seconds.

Viewing the Current Status of IP Service Activator

IP Service Activator provides the following status views:

- The statistics summary pane summarizes the status of PHB groups, policy rules, CCCs, and VPNs and can be displayed as a dockable toolbar in any window.
- The system statistics window provides a single-screen status summary: system components, policy rules, PHB groups, VPNs and CCCs, and network devices.

To view the statistics summary pane:

- From the **View** menu, select **Statistics Summary**.

  For PHB groups, policy rules, VPNs, CCCs and driver scripts the pane displays the number of each object with the following status:

  – **Inactive**: Object has been created but has not been propagated to the network processors.

  – **Disabled**: Object has been disabled.

  – **Active**: Object has been propagated to the network processors.

  – **Conflict**: Object conflicts with an existing object or has failed validation.

  – **Rejected**: One or more of the objects failed to install.

  – **Failed**: Network processor experienced a failure.

  – **Installed**: Object that has been installed on the designated device.

  – **Mismatched**: Configuration in IP Service Activator for this concrete does not match the configuration on the designated device.

  – **Uninstalled**: Object has been uninstalled. This can mean that the object is not currently active on the device (for example, because the date and time limitations do not apply at present) or the object has been deleted or disabled by a user.

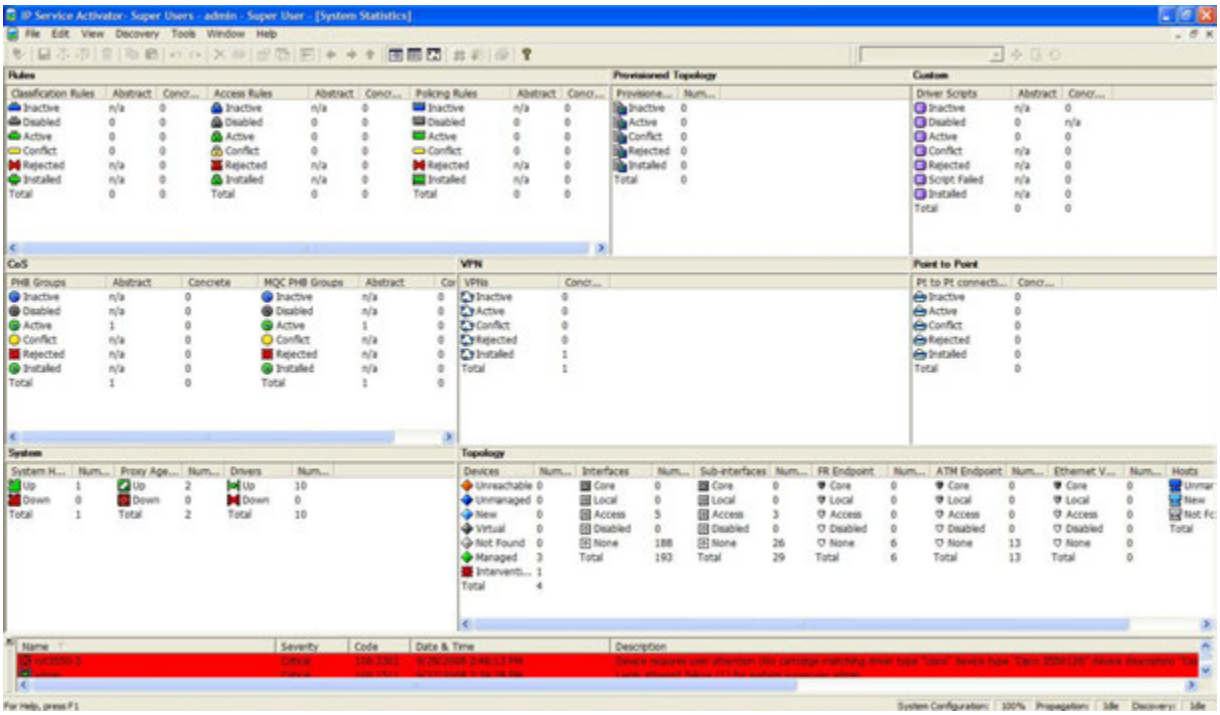
  – **UninstallFailed**: Object remains on the device, and its concrete is disabled in the GUI. (A fault may be raised.)

To view the System Statistics window:

- From the **View** menu, select **System Statistics**.

  The System Statistics window is displayed in the figure below.

Figure 6–2 The System Statistics Window



- For current classification rules, access rules, and policing rules, the window indicates the number of abstract (parent) and concrete rules that are **Inactive**, **Disabled**, **Active**, **Changed**, **Conflict**, **Rejected,** or **Installed**.
- For PHB groups, the screen indicates the number of abstract (parent) and concrete rules that are **Inactive**, **Disabled**, **Active**, **Changed**, **Conflict**, **Rejected**, or **Installed**.
- For VPNs and CCCs, the screen indicates the number of concrete VPNs or CCCs that are **Inactive**, **Active**, **Conflict**, **Rejected**, or **Installed**
- For system components (hosts, network processors, and cartridges) it indicates the number that are **Up** or **Down**.
- For devices, it indicates the number that is currently in each state: **Unreachable**, **Unmanaged**, **New**, **Not Found**, **Managed**, or **Intervention Required**.
- For interfaces, it indicates the number of interfaces whose role is **Core**, **Local**, **Access**, **Disabled,** or **None**.

  The Interfaces pane shows the number of interfaces or sub-interfaces assigned to each role. If an interface has no sub-interfaces, its role assignment is added to the count. If an interface has one or more sub-interfaces, each sub-interface role is counted and the role of the parent interface is not.
- For hosts, it indicates the number that are **New**, **Unmanaged,** or **Not Found**.

Permitting GUI Access to the Network Processor

Perform this procedure only if there is a firewall between the IP Service Activator GUI and the Network Processor server. Use this procedure to allow GUI access to the CORBA port assigned to the Network Processor, so the GUI can send device configuration audit results to the Network Processor.

The device configuration audit is performed in the expert GUI, on the Device dialog, Audit page. For more information about running a device configuration audit, refer to the IP Service Activator Online Help.

If you do not perform this setup procedure before initiating an audit on any cartridge-driven device, the GUI returns an error message stating it is unable to connect to the Network Processor.

To permit access from the GUI to the Network Processor:

1. Work with your network administrator to perform the following tasks:
   - Choose a port number for Network Processor GUI connectivity.

     Ensure your choice of port does not conflict with ports used by other network services running on the Network Processor servers.
   - Open the firewall at the chosen port.
2. Perform this step on each server where the Network Processor component is installed:
   - Start the Configuration GUI. See "Running the Configuration GUI".
   - Click **BaseHost**, **Network Processor Framework** and then **Common**.On the left-hand side panel, edit the ORBServerPort parameter.
   - Click **Validate Configuration**.
   - Click **Commit to Host**.
   - Restart the component manager to implement the change. For procedure details, refer to "Starting IP Service Activator Components".

Transaction Troubleshooting

When a transaction fails, helpful fault information is provided in *Service_Activator_Home*/**DebugLogs/System_Logger.log**.

You can review fault messages in the ReasonForFailure attribute for the failed transaction by way of the integration manager CLI. (Log into the CLI and put the failed transaction into context.) For more information, refer to the *IP Service Activator OSS Integration Manager Guide*.

Auditing

In IP Service Activator, you can view the configuration running on a device. For an in-depth analysis of the configuration, you can perform a device audit or a per-service audit.

> **Note:** Performing an audit is an administrator function.

The audit feature identifies discrepancies between an IP Service Activator configuration and the corresponding configuration currently on the device.

Viewing the Configuration on a Device

To view the configuration currently running on a device, you can use the snapshot option, as follows:

```
auditdevice -snapshot -all | <IP_address_of_device> | <device_name_in_object_
```

```
model>
```

Examples:

```
auditdevice -snapshot -all OR auditdevice -snapshot 10.13.4.115
```

The snapshot option audits the router and creates text file (*ip_address*.txt) that contains the router configuration and places it in the directory named "SnapShotDir". This option is used for internal purposes. In addition, the tftp server picks up the snapshot text file from the **SnapShotDir** directory.

Running a Device Audit

This section describes how to perform a command-line version of the device audit, where the report is stored, and how to read the report.

You can also perform a device audit on a cartridge-managed device from the **Device properties - Audit/Migrate page** in the GUI. Results are displayed in the GUI, but not stored. For details, refer to the Online Help procedure for running a device audit.

To perform a device audit from the server:

1. Telnet to the machine currently running the Network Processor and execute the following command:

   ```
   auditdevice <IP address of device> | <device name in object model>
   ```

   For example:

   ```
   auditdevice 10.13.4.115 OR auditdevice rot10008-1
   ```

   The completed audit file appears in *Service_Activator_Home*/**AuditTrails/reports**. The audit file name format is **audit.***IP address of device.date-time***.xml**.

   To view a device audit report, open the audit XML file in a web browser; the appropriate stylesheet automatically loads. The stylesheets reside in *Service_Activator_Home*/**AuditTrails/reports**. Device audits use the stylesheet **deviceaudit.xsl**.

   If you cannot access the directory *Service_Activator_Home*/**AuditTrails/reports** directly from your web browser, you must copy the xsl and audit xml files to your local machine to view the reports.

2. Analyze the reports to see if there are any configuration discrepancies that need to be corrected. See "Audit File Color Coding".

Device Audit Report Example

The following is an example of an audit report for Cisco:

Audit report for 10.13.4.115 (cisco, rot10008-1)

```
ip route 38.1.1.0 255.255.255.0 Serial2/0/1.1/3/1/1:1.21

interface Serial2/0/1.1/3/1/1:1.21 point-to-point
description test
bandwidth 128
ip address 41.6.1.1 255.255.255.248
frame-relay interface-dlci 21
```

Running a Per-service Audit

You can perform an audit on an individual service of a device, without running a full device audit.

To perform a service audit:

1. Retrieve the external IDs of the concretes for the services you want to audit:
   - Log in to the OSS Integration Manager command line interface.
   - Run the command:

     ```
     find <device ID> concreteobject:"*"
     ```

   - Run the following command and look for the external ID value:

     ```
     cat <concrete ID>
     ```

   - Repeat as necessary to retrieve all external IDs of the concretes involved in the service.
2. Telnet to the host currently running the Network Processor and run the following command:

   ```
   auditdevice -filter <external ID of concrete> <IP address of device>
   ```

   For example:

   ```
   auditdevice -filter 26890 10.13.4.115
   ```

   For the value <external ID of concrete>, you can list multiple values, separated by commas.

   The completed audit file appears in *Service_Activator_Home*/**AuditTrails/reports**. The audit file name format is **audit.<IP address of device>.<date-time>.xml**.

   To view a device audit report, open the audit xml file in a web browser; the appropriate stylesheet automatically loads. The stylesheets reside in *Service_Activator_Home*/**AuditTrails/reports**. Device audits use the stylesheet **deviceaudit.xsl**.

   If you cannot access the directory *Service_Activator_Home*/**AuditTrails/reports** directly from your web browser, you must copy the XSL files to your local machine to view the reports.
3. Analyze the reports to see if there are any configuration discrepancies that need to be corrected. See the topic "Audit File Color Coding".

   When blue text appears in the audit report, it indicates that conflicting configuration exists on the device. This is usually due to configuration added manually, unknown to IP Service Activator. You can control whether to display or hide manual configuration in the audit report for a service audit. By default, the manual configuration is not displayed in the audit report.
4. To display the manual configuration associated with a service, add or modify the following attribute in the **default.properties** file:

   ```
   suppressManualConfigInPerServiceAudit = false
   ```

   When set to false, any manual configuration is displayed in the Per Service Audit report.
5. To suppress reporting the manual configuration associated with a service, set the value of the attribute to true.

Note: When a per-service audit is initiated with the suppressAuditFeedback property in the network processor **default.properties** file set to false; the concrete audit states, and device audit state will be updated as in a full audit of the device. The generated per-service audit report is filtered using the IDs in the filter list.

Audit File Color Coding

The audit file displays text in different colors to indicate whether configuration discrepancies have been found. Table 6–13 displays the definitions of the audit file color coding when displayed in a browser using the style sheet.

Table 6–13 Audit File Color Coding

Status of the command line in the audit file	Command Re-issue View	Audit View
NORMAL: No discrepancies exist between the device and the IP Service Activator device model.	Black	Black
MISSING: Configuration command delivered by IP Service Activator is missing from the device.	Red	Red: Bold & Italic
CHANGED: The configuration is changed. The value that IP Service Activator configured on the device and the actual value now on the device are both displayed in red.	not applicable	Red: bold & italic
CONFLICT: Conflicting configuration exists on the device. It may be due to manual configuration, or commands that are in the wrong sequence, or other cause.	Blue	Blue: italic
POTENTIAL: Some commands may be marked as undisplayable in the Audit template XML file, even though they were sent by IP Service Activator.	Purple	Black: bold & italic
UNMANAGED	not applicable	Gray
WRONG ORDER: The command order in IP Service Activator is different from the command order on the device.	not applicable	Black with Yellow background

Note about CTM-generated configuration: Configuration Template Module commands are not audited by IP Service Activator. IP Service Activator device audits may display these commands as manual configuration, as if they were provisioned outside of IP Service Activator. As such, they may appear in blue (conflict) text in the audit reports.

Audit Synonyms

Audit synonyms can improve the success rate of a device audit, for devices that display commands differently from IP Service Activator.

Description

The audit compares IP Service Activator commands to the commands displayed on the device. If the two sets of commands match, the audit passes. If the two sets of commands have discrepancies, the audit fails.

Some device and OS combinations display commands in a slightly differently format than IP Service Activator. This causes the audit to fail even though the commands are equivalent. In this case, you can specify synonyms for these commands. The use of

audit synonyms applies regular expression "match and replace" (substitute) directives that allow the audit comparison to determine if two commands are equivalent. Synonyms do not appear in the audit results; they are only used at the time of comparison to see if two commands are equivalent. You can apply synonyms to the IP Service Activator-expected commands, or to the commands displayed on the device, or to both.

Each command can have only one synonym made up of one or more match and replace criteria. Each synonym can match only one command within the scope of a single audit template.

System synonyms are delivered with the product software. However, you can redefine system synonyms or create new synonyms.

Loading Audit Synonyms

When the network processor starts, it loads synonym files from two locations:

1. System synonyms are loaded from the file **synonyms.xml** located in the cartridge.
2. Custom synonyms are loaded from valid synonyms files in the directory:

   Service_Activator_Home/**Config/networkProcessor/Custom/AuditSynonyms**

   A valid synonyms file is one that conforms to the synonyms schema; its filename is not relevant.

Map folders in the synonym files are keyed by the audit template.

Maps in the synonyms files are keyed by the id. Any maps loaded later that have the same key will replace existing maps.

> **Note:** Maps for all cartridges are loaded into the same namespace. This allows maps to be shared. However, if you do not want to share a map, then you must name it uniquely so it does not *unintentionally* overwrite a previously loaded map.

Sample Synonyms File

```
<?xml version="1.0" encoding="UTF-8"?>
<synonyms xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.metasolv.com/serviceactivator/synonyms
synonyms.xsd"
    xmlns="http://www.metasolv.com/serviceactivator/synonyms">
    <mapFolder>

<auditTemplate>file://com/metasolv/serviceactivator/cartridges/cisco/units/cu1/aud
itTemplate.xml</auditTemplate>
        <mapGroup>
            <cmdRef>cisco.interface.rate-limit</cmdRef>
            <mapRef>set-mpls-exp-imposition-transmitNormalizer</mapRef>
        </mapGroup>
    </mapFolder>
    <map>
        <id>set-mpls-exp-imposition-transmitNormalizer</id>
        <mapCurrent>
            <match>(set-mpls-exp-imposition-transmit)</match>
            <replace>set-mpls-exp-transmit</replace>
        </mapCurrent>
    </map>
</synonyms>
```

Understanding How Synonyms Work

This section describes the elements in a synonyms file. For an example, see "Sample Synonyms File".

- **<mapFolder>**: The map folder consists of an audit template URL and a number of map groups. You can have multiple map folder entries within a synonyms file to organize your synonyms for reuse across multiple capabilities files.
  - **<auditTemplate>**: The map folder points to the URL of an audit template file which lists all commands applicable to a specific capabilities file.
  - **<mapGroup>**: Each map group consists of a command reference and a map reference that apply to one synonym. The synonym is applicable when the specified audit template file is in scope for the executing capabilities file.
  - **<cmdRef>**: The command reference value is the **<commandID>** defined in the audit template file for the command. It is the first indication during runtime that a synonym exists for the command.
  - **<mapRef>**: The map reference value is a pointer to the associated map ID listed later in this synonyms file, where the synonym match/replace code is defined.
  - **Note: <mapRef>** can reference map elements in any of the currently loaded synonym files. This is useful when referencing maps in the system synonyms file, but may have undesirable effects if referencing maps defined to be used with other audit templates. Prevent this problem by carefully defining command IDs. (See "Creating a Custom Synonyms File".)
- **<map>**: The map is the basic synonym element, and consists of:
  - **<id>**: identifies the map pointed to by the map reference **<mapRef>** in the Map folder.
  - **<mapCurrent>**: matches the command format configured on the device and specifies the replacement command format.
  - **<mapExpected>**: matches the activation command format sent to the device by IP Service Activator and specifies the replacement command format.
  - You can have multiple **<mapCurrent>** and **<mapExpected>** entries within a map to produce the resulting synonym.

Creating a Custom Synonyms File

- When system synonyms need modification or enhancement, you can modify entries (on a per-command basis) in the custom synonyms file, located in the following directory:

  Service_Activator_Home/**Config/networkProcessor/Custom/AuditSynonyms**
- If one does not already exist, you can create a custom synonyms file from a copy of the system synonyms file located in the sample registry directory for each cartridge. For example, for the Cisco cartridge, go to:

  Service_Activator_Home/**Config/networkProcessor/ciscoSampleRegistry**
- To define synonyms for a command, start by defining a **<mapFolder>** that references the appropriate audit template. Next, define within it a custom **<mapGroup>** element with a **<cmdRef>** that references the command ID. Next, define the **<map>** element. Then add a map reference for the map element.

- The custom **<mapGroup>** must reference all applicable **<map>** elements to produce the correct synonym for a command. In addition to adding a new map for a command in the custom synonyms file, if a map exists for the command in the system synonyms file and it still applies, you must add a **<mapRef>** in the custom **<mapGroup>** to reference the corresponding **<map>** element in the system synonyms file.

Synonym Tools

There are two synonym tools available: **auditSynonym** and **npAdmin**.

auditSynonym

You can use the **auditSynonym** tool to help you develop synonyms. It uses the perl5 regex engine.

```
UNIX: bin/auditsynonyms <regex> <substitution> <input>
Windows: Program\auditsynonyms.bat <regex> <substitution> <input>
```

Example 1:

```
> Program\auditsynonyms.bat "(black)" "red" "any value black"
Output:
Regex:  (black)
Subst:  red
Input:  any value black
Output Perl5: any value red
```

Example 2:

```
Regex:  (threshold-type (?:(?:never)?(?:immediate)?(?:average
[0-9]*)?(?:consecutive [0-9]*)?)(?:xofy [0-9]+ [0-9]+)?) (threshold-falling
[0-9]+) (action-type (?:trapOnly)*(?:trapAndNvmt)*)?
Subst:  $3 $1 $2
Input:  rtr reaction-configuration react Timeout threshold-type average 11
threshold-falling 2 action-type trapAndNvmt
Output Perl5: rtr reaction-configuration react Timeout action-type trapAndNvmt
threshold-type average 11 threshold-falling 2
```

npAdmin

You can use the **npAdmin** tool to force the network processor to reload both the synonyms and audit template files:

npAdmin.sh reload_synonyms

You can also use the **npAdmin** tool to display the current synonyms in effect on a network processor:

npAdmin.sh display_synonyms

About Audit Templates

Audit templates define filter patterns to be applied to commands to identify configuration of interest, and to affect their inclusion in the audit report. Audit templates also set attributes on the command results, which, when viewed using a stylesheet, affect how they are displayed to the IP Service Activator user.

The **auditTemplate.xml** file is located in the sample registry directory for each cartridge. For example, for the Cisco cartridge, go to:

Service_Activator_Home/**Config/networkProcessor/ciscoSampleRegistry**

Audit templates for base cartridges have two command sets. Audit templates for service cartridges have only one command set, the one described as **2nd command set**.

For a service cartridge, the 1st command set is not needed; the 1st command set from the base cartridge's audit template file describes the commands that the network processor must issue to the device to retrieve the entire configuration of the device for the base cartridge, and all of the service cartridges that extend the services of the base cartridge.

- 1st command set: the device commands to list a device's entire configuration
- 2nd command set: a complete set of the commands that the cartridge is capable of sending to the device

The following is an example of the structure of an audit template (**auditTemplate.xml**) file:

```
<commandSession
xmlns="http://www.metasolv.com/serviceactivator/climodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <command>
      <!--commands to show configuration-->
      <commandId>show-commands</commandId>
      <commandString/>
      <command kind="show">
         <commandId>show-commands.show_running-config</commandId>
         <commandString>show running-config</commandString>
      </command>
      <command>
         <commandId>show-commands.logout</commandId>
         <commandString>logout</commandString>
      </command>
   </command>
   <command>
      <!--context specific filter patterns toidentify configuration of interest-->
      <commandId>cisco</commandId>
      <commandString/>
      <command>
         <commandId>cisco.interface</commandId>
         <commandString>^interface</commandString>
         <command>
            <commandId>cisco.interface.description</commandId>
            <commandString>^description .*$</commandString>
         </command>
         <command>
            <commandId>cisco.interface.ip_address</commandId>
            <commandString>^(no\s)?ip address</commandString>
         </command>
      </command>
      <!--Alias command for configuration version -->
      <command configVersion="true"reportManualConfig="true">
         <commandId>cisco.alias_IpsaConfigVersion</commandId>
         <commandString>^alias exec IpsaConfigVersion \S+</commandString>
      </command>
   </command>
</commandSession>
```

When an audit is requested for the device, the network processor sends the first command set to the device to retrieve its configuration, which is filtered against the commands in the second command set. Commands that match those configured by

the cartridge (that is, those listed in the second command set) are converted into a CLI document.

The network processor then invokes the annotated device model to CLI transform from the cartridge on the last pushed (or persisted) device model (which matches the IP Service Activator internal representation of what is configured on the device).

A comparison is made between the two CLI documents. This comparison can highlight discrepancies including:

- commands sent to the device that are now missing
- commands added to the device that are not in the network processor device model (manually configured outside IP Service Activator)
- commands that are in the wrong order on the device. (In some cases command ordering affects how the device works.)

Understanding audit template elements

The audit template referenced by each capabilities file, and by each synonyms file, lists all commands that can be applied by the cartridge.

- Each command has a **<commandString>** and an associated **<commandId>**. The command ID can imply its command string, but should use a hierarchical structure, as in the example.
- A **<commandId>** is an identifier that tells IP Service Activator that the command has a synonym.
- A **<commandId>** must be unique within an audit template. The same **<commandId>** can be used for the same command in different audit templates that reuse the same synonym.
- Each unique version (format) of a command string should have a unique command ID. For example, the **<commandId>** should be different in different audit templates when the same command requires a different synonym (because of OS display differences).
- The **<commandString>** is the actual text of the command. Commands and their parameters can be hierarchically defined in the Audit template, as shown in the previous sample.

Audit Template Command Attributes

Audit template attributes are used to accommodate variations in device responses and to customize the audit report. You can apply these attributes to each individual <command> element to tell the audit to treat this command in a particular manner.

For example, when you read device configuration, the device could:

- display a command in lower case where IP Service Activator provisioned the device in upper case
- display commands indented incorrectly (resulting in wrong context interpretation)

Attributes can be used to ensure the expected command format is displayed in the audit report.

You can use attributes to indicate that specific commands need to follow one another in a particular order.

Attributes

- kind=<string>
- configVersion=<boolean>
- ordered=<string>
- autoIndentUtil=<string>
- ignoreCase=<string>
- reportManualConfig=<boolean>
- brokenIndentRulesOffset=<int>

kind=<string>

The kind attribute affects how the command is used to match commands on the device. Table 6–14 lists the effect of the kind attribute value:

Table 6–14 Using Kind Attribute Values

Kind Attribute Value	Attribute Value Effect
ALWAYSIGNORE	Excludescommand from the audit report when found in any context.
IGNORE	Excludes command from the audit report when found in this context.
IGNORE_CASE	Performs a case insensitive comparison with command on device.
IGNORE_CHARS	Performs a comparison with commands on device ignoring characters that follow the attribute (e.g. <command kind="IGNORE_CHARS -">. A dash (-) appears in the device configuration, however, you want the audit to ignore it.
CHECK_DEFAULT	Performs a comparison with commands on the device marking missing commands as "potentially" missing. This is used for special handling in cases where devices do not display default values in a command string.
PARTIAL	Performs a comparison with commands on the device taking the command element name (as depicted in the audit template) as part of the full command. This allows the audit to determine equality between IP Service Activator provisioned commands and those seen on the device using a partial comparison only.
NO_ORDER_PARAMS	Performs a comparison where the command has to match the regular expression exactly up to where it is explicitly defined in the audit template, but the rest of the command (.* is assumed at the end of the regular expression) is compared regardless of the actual order. So if the same attributes appear out of order but are matching, it results in a match.
SUBSET	Performs a comparison where the command has to match the regular expression exactly up to where it is explicitly defined, but the rest of the command (.* is assumed at the end of the regular expression) must be included in the configured command in order to match; in other words, it matches if it is a subset of the actual list that is configured.
HIDE_SECURE_INFO	Performs a comparison in the same manner as PARTIAL, but is specifically intended for secure information like passwords. This avoids showing the secure information even when using showAll.

Effect of Kind Attribute on Audit Report

The audit template command kind attribute affects the inclusion of the command, and its kind attribute value in the resulting audit report. Table 6–15 shows the effect of the kind attribute on the resulting audit report:

Table 6–15 Effect of Kind Attribute on Audit Report

Cmd in IP Service Activator	Cmd on Device	Command in Audit Template	"Kind" in Resulting Audit Report showAll=false (IPSA)	"Kind" in Resulting Audit Report showAll=true (CM)
C1	C1	Present	NORMAL	
C1	C1	Present + kind=CHECK_ DEFAULT	NORMAL	
C1	C1	Present + kind=IGNORE	*exclude*	UNMANAGED
C1	C1	Not Present	*exclude*	NORMAL
C1	C1′	Present | Present + kind=PARTIAL	CHANGED	CHANGED (Missing:Conflict)
C1	C1′	Present + kind=PARTIAL_ DISPLAY_ALL	MISSING	CHANGED (Missing:Conflict)
C1	C1′	Present + kind="CHECK_ DEFAULT"	CHANGED	CHANGED (Potential:Conflict)
C1	C1′	Present + kind="IGNORE"	*exclude*	UNMANAGED
C1	C1′	Not Present	*exclude*	CHANGED (Missing:Unmanaged)
C1	missing	Present	MISSING	
C1	missing	Present + kind="CHECK_ DEFAULT"	POTENTIAL	
C1	missing	Present + kind="IGNORE"	*exclude*	*exclude*
C1	missing	Not Present	*exclude*	MISSING
--	C1	Present	CONFLICT	
--	C1	Present + kind="CHECK_ DEFAULT"	POTENTIAL	
--	C1	Present + kind="IGNORE"	*exclude*	UNMANAGED
--	C1	Not Present	*exclude*	UNMANAGED

Note: C1 means the command is the same, but some of its arguments are different.

configVersion=<boolean>

Set configVersion to **true** on the command that sets the IP Service Activator configuration version. The IP Service Activator configuration version is a timestamp of when the configuration was last modified by IP Service Activator.

ordered=<string>

The ordered attribute lets you control whether or not the sequencing of commands is relevant when determining if discrepancies exist between the device and the IP Service Activator device model. Define a string value and set the ordered attribute value to the same string value for all commands at the same nesting level for which the sequencing is relevant. The correct sequence of the commands is the order in which they appear in the audit template.

Example

To specify that if commands "cmd B", and "cmd D" both appear under "cmd X", then "cmd D" must appear after "cmd B", set ordered="OrderBD" for both "cmd B", and "cmd D".

In this case, no discrepancies are found, because B and D appear in the correct sequence relative to each other:

```
cmd X
cmd A
cmd B
cmd C
cmd D
cmd E
```

In this case, discrepancies will be found since B, and D do not appear in the correct sequence relative to each other:

```
cmd X
cmd D
cmd A
cmd B
cmd C
cmd E
```

autoIndentUntil=<string>

This attribute reports to the audit that every cmd under the current cmd is a child cmd until "<string>" is detected; after that, the context of the current cmd is exited. Use this command to deal with bad indenting.

Example

```
autoIndentUntil="cmd D"
cmd A
cmd B
cmd C
cmd D
cmd B, C, D are support be children of cmd A
```

reportManualConfig=<boolean>

The reportManualConfig attribute lets you control whether a command is reported as a manual configuration. By default, global commands are not reported as manual configurations, but nested commands are.

brokenIndentRulesOffset=<int>

Reports to the audit that this cmd has a bad indent and that it should be corrected to absolute value X. This attribute deals with bad indenting and comes into play when CLIParser breaks out of context.

Example

for cmd D

brokenIndentRulesOffset="2"

```
cmd A
cmd B
cmd C
cmd D
cmd E
```

In this example, cmd D is supposed to be a child of cmd A, however its relative position shows it is out of context for cmd A. The brokenIndentRulesOffset attribute is used when audit interrogates cmd D and is about break out of cmd A context, the absolute position of cmd D is then reset to ensure that cmd D is a child of cmd A or peer to cmd C, E, and so on.

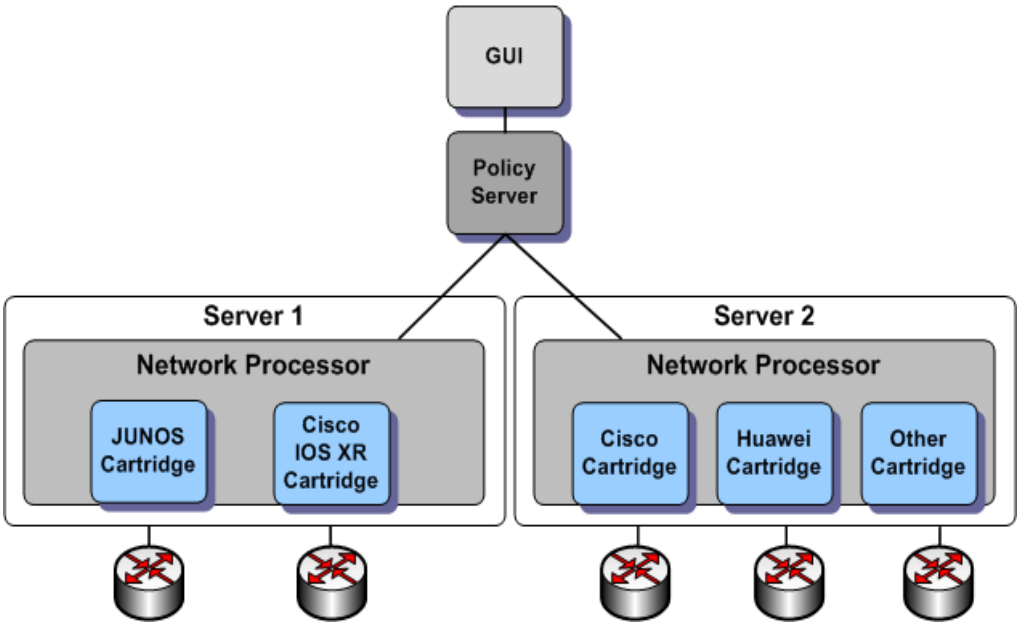
7

Monitoring and Managing the Network Processor

This chapter provides feature, administration, maintenance, and troubleshooting information for the Network Processor and its cartridges.

The Network Processor/Cartridge combination is implemented in a Oracle Communications IP Service Activator system, as shown in Figure 7–1.

Figure 7–1 IP Service Activator System with Network Processor and Cartridges



More than one Network Processor can be installed in a distributed system, as long as each server contains only one Network Processor. A Network Processor can have multiple cartridges.

Network Processor

The Network Processor uses Activation Cartridges that include XML-based vendor-specific and service-specific definitions for a number of device types.

Supporting an enhanced stateful activation model, the Network Processor uses a persistent store in the Oracle Database to keep a record of the details and state of services deployed on devices.

The Network Processor component allows the introduction of service support on new device types or variants of existing types. You can develop new cartridges and configuration policies using the IP Service Activator SDK.

Multiple Network Processors can each load an instance of the same cartridge, thereby allowing to scale up to a higher number of managed devices. The Network Processor and cartridge reduces the system restart time, and improves overall system stability.

The Network Processor provides multi-vendor capability, with vendor-specific pieces delivered as cartridges.

The Network Processor runs under the control of the Component Manager. The Component Manager is responsible for starting, stopping, and restarting the Network Processor, and for reporting component failures.

Network Processor and Cartridge Components

The main components of the Network Processor and cartridge are shown in Figure 7–2.

Figure 7–2 Network Processor, Cartridge Components, and CLI Commands to Device

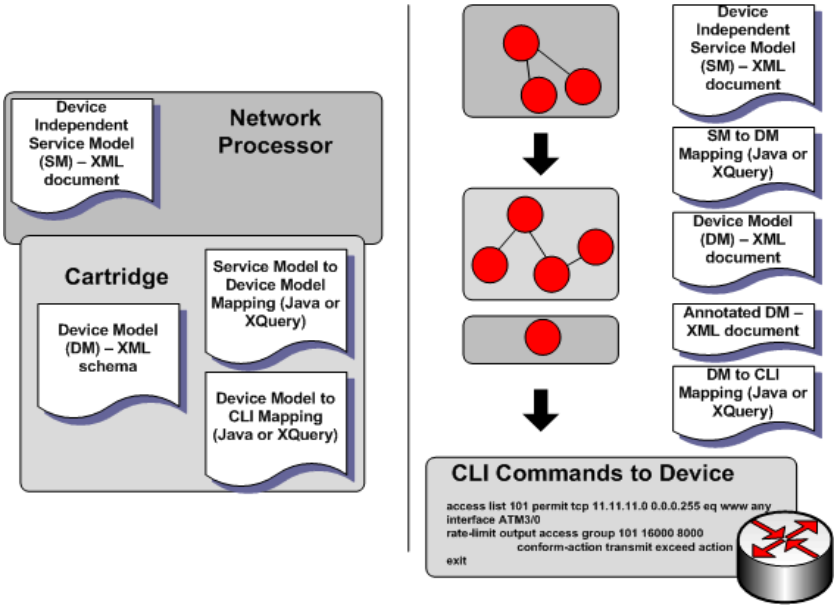
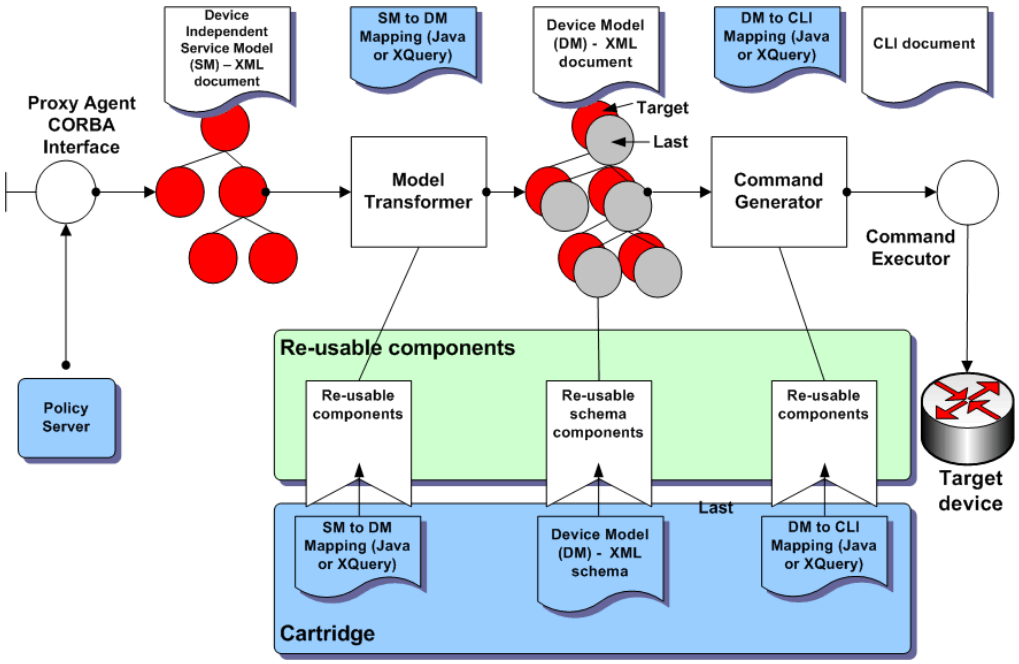


Figure 7–3 illustrates the flow of data through the Network Processor.

Figure 7–3 Network Processor Data Flowchart from the Interface to the Target Device



The main components are described as follows:

- A **Network Processor** is a processing component that performs the conversion of user changes into a set of CLI commands for delivery to devices.
- A **Service model** is a device-independent representation of service objects and their relationships. It resembles the information model configured by the user.
- A **Device model** is a device-specific XML document derived from the service model. It contains a set of data elements that defines the complete device state.
- A **Cartridge** is a vendor-specific implementation of a set of services for a given family of device types running the same operating system. It may be composed of multiple capabilities units.
- A **Capabilities file** defines an instance of specific, concrete, complete behavior (including command syntax) for a service or a set of services. It is applicable to device/OS combinations with identical CLI behavior. For more information see, "Capabilities and Options". A capabilities file contains:
  - a service model transformation (Java or XQuery script).
  - a device model schema definition (XML Schema). It governs the structure of the Device model.
  - a command generation (Java or XQuery script). The output is the CLI, the set of commands specific to the target device type.
- The **Cartridge registry** maps device type/OS combinations to capabilities files. When a transaction arrives, the Network Processor looks in the cartridge registry file (**MIPSA_registry.xml**) and selects a capabilities file based on the supported device type and OS.

  If you have a device type/OS combination that is not present in file **MIPSA_registry.xml**, you must create a new cartridge registry file, as outlined in "Creating or Editing a Cartridge Registry File" or "Creating or Editing a Cartridge Registry File - Juniper Example".

Network Processor Operation

The Policy Server takes user input and computes all its effects on the modeled network. For each device affected, if the device is managed by a Network Processor, the Policy Server sends the device-specific changes to the Network Processor. The Network Processor assembles an abstract service level model for the device, called the Service Model.

The Network Processor matches the device's type and IOS information against the Registry files in the cartridges to identify the specifics of handling this particular device. The Network Processor uses the specifications for transformation, validation, capabilities and options (among others) in the matching Registry entries to convert the device Service Model into one or more target Device Model documents. These documents represent the services as configured on the device.

The Network Processor compares the target documents with the last saved Device Model documents for this device and generates an annotated document that describes the changes. It again uses the matching Registry entries to convert this annotated document into commands - which are then dispatched to the device and recorded in the audit log.

If command delivery is successful, the target Device Model documents and the computed Service Model used are saved to the database, replacing the older saved documents. For a given device, there can be a maximum of one "current" set of documents, as identified by the document versions in the database.

If command delivery fails for whatever reason on the device, the Network Processor rolls back all commands delivered so far. It then isolates the failing command and attempts to undo the last change that triggered this command (and replace it with the last known successful state). The Network Processor then recomputes the commands to be delivered to the device and attempts the delivery again. This cycle is repeated until either all configuration changes have individually succeeded or failed. At that point, Network Processor raises faults in the Policy Server and GUI to indicate failure and provides more information on what failed and why.

Implementing New Services Through the Network Processor

Before pushing a new service, an audit of the device should always be performed to make sure that there is no conflicting configuration on the device.

Tolerance for Manual Configuration on Devices

The Network Processor changes only those configuration elements on the device that are indicated by the user input.

The Network Processor changes existing configuration on the device only if the device response is a success message, one of the messages in the **SuccessMessages.xml** file. A success message indicates that the device does not see any conflict between the existing configuration and the new configuration sent to the device.

When the device response is not a success message, IP Service Activator rolls back the transaction on the device. It raises a fault, records the configuration conflict in the audit file (**np**<*cartridgeName*>**.audit.log**), and marks the device state as **Intervention Required**, for decision/action by the administrator.

Adding a New Success Pattern

To add a new success pattern, edit the **SuccessMessages.xml** file.

1. Navigate to *<installDir>***/ServiceActivator/Config/networkprocessor/com/metasolv/serviceactivator/networkprocessor/**
2. Edit **SuccessMessages.xml** and insert the new success pattern at the end of the required section. For example:

```
<!--@ CISCO MESSAGES -->
...
<cmd:successPattern>
<!-- -cisco : message after "no no ip route"  -->
<cmd:pattern>(?s).*%No matching route to delete*</cmd:pattern>
</cmd:successPattern>
```

3. Save the file.

Configuring the Network Processor

This section describes how to configure the Network Processor.

Configuring Performance Characteristics of the Network Processor

Two performance parameters of the Network Processor are configurable. The ability to configure the **memory cache size** for device model and service model data enables better control over the trade-off between resources and performance. The multi-threaded implementation of the Network Processor means that one Network Processor installed on a server can fully utilize the server CPU without limiting throughput. The **number of simultaneous device transactions** is configurable.

This section provides procedures for tuning two Network Processor performance parameters:

- the size of the Network Processor cache
- the number of simultaneous device transactions allowed

Both parameters affect CPU throughput, so it is suggested to adjust the two values to provide optimum balance. You can adjust these parameters by editing values in the **default.properties** file, located in directory: **/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor.**

Setting the Network Processor Cache Size

Each device has a service model and a device model associated with it. The Network Processor holds in cache memory the current service model and device model data for a configurable number of managed devices. Configuring the cache size helps optimize system performance. Increasing the cache size may speed up system performance because the system refers to data held in memory, rather than in the database. Setting too large a cache size may cause performance problems on other memory-intensive activities, so seek the right balance when you set the cache size.

There are two caches, one for Service Model data, and one for Device Model data. To configure the network processor cache size memory limit, see "Configuring Memory Limits for Network Processor Cache".

Configuring the Number of Simultaneous Device Transactions for Network Processor

After the Network Processor is installed, you can configure the number of simultaneous device transactions to optimize system performance. The maximum number of "threads" (maxThreads) depends on the number of CPUs installed and their processor speeds.

The default values are minimum threads = 1 and maximum threads = 20. You can edit these values in the **default.properties** file.

Specify the thread count based on performance of the memory-intensive activities. Oracle recommends setting a maximum of 4 threads per CPU. The formula is as follows:

maxThreads = 4 * number of CPUs

Configuring Queue Priority at Network Processor Startup

You can set the priority for pushing configuration to devices on Network Processor startup or restart, to improve the Network Processor availability to process new work. The value of the **queuePriority** parameter determines which commit queue is used on the initial push of configuration to a device on Network Processor restart. The values of this parameter behave as follows:

- **auto**: prioritizes the initial configuration push based on the actual content of the device's service model: If the model has changes to push to the device, then it is assigned to the high priority commit queue. If there are no changes, then it is assigned to the low priority commit queue.
- **high**: puts the initial configuration push to all devices managed by a Network Processor in the high priority queue. This results in lower Network Processor availability.
- **low (default)**: puts the initial configuration push to all devices managed by a Network Processor in the low priority queue. This results in higher Network Processor availability.

Syntax: queuePriority = [auto | high | low]

Edit this parameter in the **default.properties** file in the directory: *Service_Activator_Home*/**Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor.**

Creating or Editing a Cartridge Registry File

If you have a device type/OS combination that is not present in the cartridge registry (**MIPSA_registry.xml**), you must create a new file or a new entry in an existing file by following the steps below.

> **Note:** Do *not* modify **MIPSA_registry.xml** in the cartridge.

To create or edit a Cisco cartridge registry file:

`Go to: Service_Activator_Home\Config\networkProcessor.`

1. Create a file called **MIPSA_registry.xml** if it does not already exist. If the file does exist, open it for editing.
2. Go to:

*Service_Activator_Home***Config\networkProcessor****Device_Type***SampleRegistry**

where *Device_Type* is the device name, such as Cisco.

3. Locate the **sampleMIPSA_registry.xml** file for that cartridge.
4. Open the **sampleMIPSA_registry.xml** and locate a capabilities file entry similar to the device type/OS combination you have. Following is an example of a capabilities file entry.

   Capabilities file entry:

```
<cartridgeUnit>
<name>com.metasolv.serviceactivator.cartridges.cisco.units.cu1.3640.12.2(8)T4</
name>
    <driverType>cisco</driverType>
    <deviceType>Cisco 3640</deviceType>
    <osVersion>12.2(8)T4</osVersion>
<smToDmQuery>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/sm2dm.xq<
/smToDmQuery>
<dmValidation>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/dmValida
tion.xq</dmValidation>
<dmToCliQuery>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/annotate
dDm2Cli.xq</dmToCliQuery>
<capabilities>com/metasolv/serviceactivator/cartridges/cisco/capabilities/cisco
_3640_12.xml</capabilities>
<warningMessages>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/warni
ngMessages.xml</warningMessages>
</cartridgeUnit>
```

5. If you are editing an existing **MIPSA_registry.xml** file, copy **only** the capabilities file entry and paste it in your file.

   For information about editing the **MIPSA_registry file** for a Cisco IOS cartridge, see the topic about generating the registry file in *IP Service Activator Cisco IOS Cartridge Guide*.

   If you are creating a new **MIPSA_registry.xml** file, you must copy the XML header and footer in addition to the capabilities file entry and paste it in your file.

   XML header:

```
<?xml version="1.0" encoding="UTF-8"?> <!-- -*- nxml -*- -->
<cartridgeUnits xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.metasolv.com/serviceactivator/networkprocessor/c
artridgemanager/ ../../networkprocessor/cartridgemanager/cartridgeRegistry.xsd"
xmlns="http://www.metasolv.com/serviceactivator/networkprocessor/cartridgemanag
er/">
```

   XML footer:

```
</cartridgeUnits>
```

6. Change the name, device type, and OS version to match your specific combination. Ensure the name you select is unique.

   Note: The only parameters you need to change are name, device type, and OS version.

7. Save the newly created **MIPSA_registry.xml** file.

8. You will reload the registry file if you restart the Network Processor. To reload the registry file without restarting the Network Processor, enter:

   ```
   npAdmin.sh reload_registry.
   ```

Creating or Editing a Cartridge Registry File - Juniper Example

The following is an example of how to create or edit a file specifically for a Juniper JUNOS cartridge.

To create or edit a Juniper cartridge registry file:

1. Copy **junosSampleRegistry** to here:

   ```
   /opt/OracleCommunications/ServiceActivator/Config/networkProcessor
   ```

2. Rename to **junos**.

3. Copy **junos.xml** to here:

   ```
   /opt/OracleCommunications/ServiceActivator/Config/networkProcessor/Custom/Registries/junos.xml
   ```

4. Navigate to this path:

   ```
   /opt/OracleCommunications/ServiceActivator/Config/networkProcessor/junos/capabilities
   ```

5. Modify this line:

   ```
   <capabilities>
         <capabilitiesEntry>
           <capsFile>junos/capabilities/junos_default.xml</capsFile>
           <appliesTo>
             <deviceTypes useRegex="true">Juniper.*</deviceTypes>
             <osVersions useRegex="true">.*</osVersions>
           </appliesTo>
         </capabilitiesEntry>
   ```

6. Update the junos_default.xml file with the following entries:

   ```
   <!-- PPP -->
       <caps:ifType>23</caps:ifType>
   ```

Producing a Device Configuration Change Log

There are two ways to produce a log of device configuration changes resulting from registry changes. The first method requires a restart of the Network Processor. The second method does not.

1. Using the Configuration GUI, set the value for the optimizeFirstCommit property to false.

2. Using the Configuration GUI, put the Network Processor into OfflineTest mode.

3. Restart the Network Processor.

4. For the changes to take effect, affected devices must be managed, if they are not already.

   Configuration changes, if any, will be written to the cartridge audit log file prefixed with |no-command-delivery|.

To apply the device configuration changes resulting from registry changes without restarting the Network Processor:

1. Using the Configuration GUI, set the value for the optimizeFirstCommit property to false.
2. Do one of the following:
   - Using the Configuration GUI, put the Network Processor into OfflineTest mode
   - Put each affected device in OfflineTest mode in the IP Service Activator GUI, and commit the transaction.
3. Use npAdmin tool to reload the registry:

   ```
   npAdmin reload_registry
   ```

4. For each device for which you want to produce an audit log of device configuration changes without applying them right away:
   - Unmanage the device, and commit the transaction.
   - Check the logs to ensure the unmanage is completed before proceeding.
   - Re-manage the device and commit the transaction.

   Configuration changes, if any, will be written to the cartridge audit log file prefixed with |no-command-delivery|
5. View the resulting audit log to confirm that any configuration changes are as expected.
6. Depending on the approach taken in step 2:
   - Using the Configuration GUI, take the Network Processor out of OfflineTest mode.

     OR
   - Put each affected device back in Online mode in the IP Service Activator GUI, and commit the transaction
7. For each device for which you want to apply the changes to the device:
   - Unmanage the device and commit the transaction.
   - Check the logs to ensure the unmanage is completed before proceeding.
   - Re-manage device and commit the transaction.

   Because the npAdmin tool reloads all cartridge registry files, if any other cartridge registry changes have been made since the last Network Processor restart or reload of the registries, they will take effect now.

   At the end of this alternative procedure, it is possible that device configuration changes, which result from the cartridge registry changes are not applied. This can happen by choice or by mistakenly assuming a device is unaffected and not unmanaging and re-managing the device to apply the changes. Such outstanding device configuration changes, which have not been applied at this time, will be applied when the next policy changes are committed to the device, or when the Network Processor is restarted, whichever comes first, as long as the device is managed and online. The new MIPSA_registry.xml file is implemented.

Applying Cartridge Registry Changes

1. To apply cartridge registry changes, first follow "Producing a Device Configuration Change Log".

2. View the resulting audit log to confirm that the configuration changes are as expected.
3. Using the Configuration GUI, take the Network Processor out of OfflineTest mode.
4. Restart the Network Processor.

   For the changes to take effect, affected devices must be managed, if they are not already. Configuration changes, if any, will be applied to managed devices affected by the registry changes.
5. The value of optimizeFirstCommit can be set back to true to improve the performance of the next Network Processor restart. See "Setting optimizeFirstCommit" for more information.

Registering Cartridges with Registry.xml

There is another way to register cartridges other than using the MIPSA_Registry. When a cartridge is installed, by extracting it to the *Service_Activator_Home* directory, it creates a sample registry configuration directory under it such as *Service_Activator_Home***Config\networkprocessor\ciscoSampleRegistry**. The name of the sample directory is <*sdk_global_cartridgeName*>**SampleRegistry**. This directory contains samples of the various configuration files that you may want or need to customize.

One of the main files that you must edit to customize any SDK registry entry is the customization registry file. A sample of this file exists in the sample directory. The name of the file is *sdk_global_cartridgeName***.xml**. To use the customization registry file to customize the SDK registries, edit the file and then copy it to the directory *Service_Activator_Home***Config\networkprocessor\Custom\Registries**. The next time the network processor is restarted, the customizations override the registry entries of the cartridge.

The name of the file, as it exists in the *Service_Activator_Home***Config\networkprocessor\Custom\Registries** directory is not important. It is the name field within the file that indicates which SDK registry the customization will edit.

The customization file must conform to the **cartridge.xsd** schema.

Note: Custom registries must be specified as .xml files as the Network Processor only recognizes custom registries with the .xml extension.

Cisco sample

```
<?xml version="1.0" encoding="UTF-8"?>
<registry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.metasolv.com/serviceactivator/networkprocessor/cart
ridgeregistry/ ../cartridge.xsd"
xmlns="http://www.metasolv.com/serviceactivator/networkprocessor/cartridgeregistry
/">
<customizations>
<name>cisco</name>
<audit>
<auditTemplate>
<auditTemplateEntry>
<auditTemplateFile>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/audit/
auditTemplate.xml</auditTemplateFile>
<appliesTo>
```

```
<deviceTypes useRegex="true">.*</deviceTypes>
<osVersions useRegex="true">.*</osVersions>
</appliesTo>
</auditTemplateEntry>
</auditTemplate>
</audit>
<messages>
<success>com/metasolv/serviceactivator/cartridges/cisco/messages/successMessages.x
ml</success>
<warning>com/metasolv/serviceactivator/cartridges/cisco/messages/warningMessages.x
ml</warning>
<error>com/metasolv/serviceactivator/cartridges/cisco/messages/errorMessages.xml</
error>
</messages>
<capabilities>
<capabilitiesEntry>
<capsFile>com/metasolv/serviceactivator/cartridges/cisco/capabilities/cisco_
default.xml</capsFile>
<appliesTo>
<deviceTypes useRegex="true">Cisco.*</deviceTypes>
<osVersions useRegex="true">.*</osVersions>
</appliesTo>
</capabilitiesEntry>
</capabilities>
<options>
<optionsEntry>
<optionsFile>com/metasolv/serviceactivator/cartridges/cisco/options/cisco_
options.xml</optionsFile>
<appliesTo>
<deviceTypes useRegex="true">Cisco.*</deviceTypes>
<osVersions useRegex="true">.*</osVersions>
</appliesTo>
</optionsEntry>
</options>
</customizations>
</registry>
</registry>
```

Customization File Entries

- auditTemplate
  - The auditTemplateEntry specifies an auditTemplateFile and the deviceTypes and osVersions that the audit template is to be used for.
  - The auditTemplateFile specifies the path to the auditTemplate file relative to the directory *Service_Activator_Home***Config\networkprocessor**.
  - When multiple auditTemplateEntry entries exist, the first matching entry is used for a device. For this reason, more specific device specifications should be placed before less specific device specifications.
  - When a custom auditTemplate entry is specified, it replaces all auditTemplate entries that may have been specified in the default registry. The custom entries are not merged with the default entries; they replace the default entries.
- auditQuery
  - The auditQueryEntry specifies an auditQueryFile and the deviceTypes and osVersions that the audit query file is to be used for.

  - The auditQueryFile specifies the path to the auditQuery file relative to the directory *Service_Activator_Home***Config\networkprocessor**.
  - When multiple auditQueryEntry entries exist, the first matching entry is used for a device. For this reason, more specific device specifications should be placed before less specific device specifications.
  - When a custom auditQuery entry is specified, it replaces all auditQuery entries that may have been specified in the default registry. The custom entries are not merged with the default entries; they replace the default entries.
  - auditQuery is only valid for specific cartridges.
- success
  - Specifies a custom success messages file. The path to the file is specified relative to the directory *Service_Activator_Home***Config\networkprocessor**.
- warning
  - Specifies a custom warning messages file. The path to the file is specified relative to the directory *Service_Activator_Home***Config\networkprocessor**.
- error
  - Specifies a custom error messages file. The path to the file is specified relative to the directory *Service_Activator_Home***Config\networkprocessor**.
- capabilities
  - The capabilitiesEntry specifies a capabilitiesFile and the deviceTypes and osVersions that the capabilities file is to be used for.
  - The capabilitesFile specifies the path to the capabilities file relative to the directory *Service_Activator_Home***Config\networkprocessor**.
  - When multiple capabilitiesEntry entries exist, the first matching entry is used for a device. For this reason, more specific device specifications should be placed before less specific device specifications.
  - When a custom capabilities entry is specified, it replaces all capabilities entries that may have been specified in the default registry. The custom entries are not merged with the default entries; they replace the default entries.
- options
  - The optionsEntry specifies an optionsFile and the deviceTypes and osVersions that the options file is to be used for.
  - The optionsFile specifies the path to the options file relative to the directory *Service_Activator_Home***Config\networkprocessor**.
  - When multiple optionsEntry entries exist, the first matching entry is used for a device. For this reason, more specific device specifications should be placed before less specific device specifications.

When a custom options entry is specified, it replaces all options entries that may have been specified in the default registry. The custom entries are not merged with the default entries; they replace the default entries.

Using Activation Retry

The activation retry feature increases the probability that IP Service Activator will successfully deliver activation configurations to network devices using the following methods:

- Connection retry: IP Service Activator retries to establish a Telnet connection if the initial attempt fails.
- Command retry: IP Service Activator retries to resume command delivery if a Telnet connection fails while an activation session is in progress.

This feature also provides a troubleshooting solution for Telnet connection failures or slow network response times, allowing you to modify the intervals that IP Service Activator uses when retrying to send configuration commands to a device.

Connection Retry

If the initial attempt to connect to a device fails, IP Service Activator retries to connect for a configurable period of time. The following properties control the interval and the maximum retry time. Default values are shown. You can configure new values in the **default.properties** file, located in directory: **/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor**.

```
# Minimum time between connection retries
minSecondsBetweenIndividualConnectAttempts=21
# Total time for attempting to connect to the device
overallConnectAttemptSeconds=60
```

After a failed connection attempt or a connection failure, IP Service Activator waits 21 seconds and then tries to connect again. Retry is repeated until 60 seconds has elapsed from the initial connection attempt, at which point IP Service Activator raises a fault and may set the device state as Unreachable. (The Total time value is measured from the first connection attempt, successful or not, not from the first retry attempt.)

This feature applies to all cartridges under the Network Processor proxy agent.

Command retry

If the IP Service Activator initial attempt to connect to a device succeeds, but the Telnet connection drops while delivering commands, the Connection retry feature tries to re-establish the connection.

If the connection is re-established, IP Service Activator issues the commands necessary to restore the context for the failing command, re-issues the command, and continues until finished. If IP Service Activator fails to reconnect, the device is put into the Intervention Required state.

When the Command retry feature has operated, the line "#Resuming Configuration after lost connection" is recorded in the Audit Trail for the device.

This feature applies to Cisco and Huawei cartridges.

Configuring Time Delays for the Network Processor

Important: Do this procedure only if advised to do so by Oracle Global Customer Support (GCS).

Time delays can be configured to allow specified amounts of time before:

- Individual commands are sent to devices.
- The post-quarantine Service Model-to-Device Model (SM-To-DM) transform begins. This delay is applied each time a quarantine of failed SM-To-DM

associations occurs. For more information about SM-to-DM mapping, see "Network Processor and Cartridge Components".

You can define the parameters for these time delays in the **default.properties** file, located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

In the file, locate and modify the following values:

```
#used to specify cmd delivery delay in milliseconds between sending cmds to the
device
cmdExecutorDelay=0
#used to specify delay in seconds between quarantine invocations
quarantineDelay=0
```

Configuring Memory Limits for Network Processor Cache

You can define these cache memory configuration parameters in the **default.properties** file, located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

In the file, you can specify the following:

- **cacheCapacity**: Sets an arbitrary total upper limit for the combined size of the service and device model caches. The minimum capacity is set to 16 KB (16384 bytes), while the upper limit is set to 90% of the maximum heap available to the JVM as specified in the -Xmx command line argument (found in the **/opt/OracleCommunications/ServiceActivator/bin/networkprocessor** startup script). If a cacheCapacity value is not set, the cacheLoadFactor is used instead to calculate the memory limit.
- **cacheLoadFactor**: Determines how much of the maximum heap available to the JVM is available to the cache. This is used in the event that cacheCapacity is not set. The default value is 0.4, or 40% of the maximum heap. The permitted range is 0.1 (10%) to 0.9 (90%). It is important to remember that this determines the total cache capacity. The cacheBalance is used to determine how much is available to the individual caches.

  Note: The above two parameters should be used only if you are, 1) An experienced user of IP Service Activator, and 2) Following engineering guidelines or are advised to do so by Oracle Global Customer Support (GCS).

- **cacheBalance**: Distributes memory resources between the device and service model caches in predefined proportions. The cached device models are expected to be larger than the corresponding cached service models, so the default value is 0.67 which means that the device model cache would occupy 0.67 * cacheCapacity, whereas the service model cache would occupy 0.33 * cacheCapacity. The limits on the balance are 0.25 and 0.75.
- **cacheTrimFactor**: Refines the size estimates for cache entries. Cache entry sizes are determined largely by calculating the sizes of the payloads retrieved from either the policy server or the backing store (cache dependent) or when the entry is written back to the backing store (currently DM cache only). The default value is 1.0, meaning that the estimated memory consumption of a device model

document that was read from the database at 100 bytes is 1.0 * 100 bytes. It is possible that the memory consumption for the document is more or less than the initial size of the document text, so the trim factor provides a mechanism for adjusting capacity usage estimates.

For example, if the actual memory consumption of a 100-character device model document is 200 bytes, the trim factor should be set to 2.0, so the estimated memory used would be 2.0 * 100 = 200.

The trim factor can be determined after a number of entries have been added to the cache. A command has been added to ComponentParameters that will instruct the specified caches to determine their own ideal trim factors based on a measurement of the sizes of the entities in the cache. The command is invoked as follows:

```
ComponentParameters [the usual invocation] -set cache trim
```

-or-

```
ComponentParameters [the usual invocation] -set cache "trime {target(s)}"
```

Using the first invocation, or the second and specifying 'all' as the target, will cause both caches to adjust their trim. Otherwise, the second form can be used to adjust individual cache trim. The new trim factor is not persisted in the properties file.

Note: The above two parameters should only be used if advised to do so by Oracle Global Customer Support (GCS).

Configuring Prefix List Ranges for Cisco Devices

Prefix lists provide a prefix-based filtering mechanism. You can define the prefix list range for Cisco in the **default.properties** file, located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

In the file, locate and modify the following values:

```
# Used by Cisco transforms
ciscoPrefixListMinSequence = 10000
ciscoPrefixListMaxSequence = 99999
```

Deleting Address Family Unicast Commands from Cisco Devices

Using the **ciscoIosXrDeleteBgpAddressFamily** property, you can configure IP Service Activator to delete or not delete **address-family vpnv4 | vpnv6 unicast** commands on IOS XR Cisco devices using BGP configuration mode when all the corresponding address-family elements are deleted from VRFs configured by IP Service Activator. This property applies to IPv4 and IPv6.

This property is set in the **default.properties** file located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

By default, this property is set to **true**, which means that IP Service Activator deletes **address-family vpnv4 | vpnv6 unicast** commands. When the property is set to **false**, IP Service Activator does not delete **address-family vpnv4 | vpnv6 unicast** commands from BGP configuration mode.

Setting the "alias ip-vrf" Command for Cisco VPNs

The "alias ip-vrf" command provides a tagging method to indicate, on the device, VRF configurations that are owned by IP Service Activator. It is an optional command for all Cisco VPN configurations. The property **VrfAliasesDisabled** in the **default.properties** file allows you to control whether "alias ip-vrf" configurations are sent to the device.

The **default.properties** file is located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

By default, the **VrfAliasesDisabled** property is set to "false", meaning "alias ip-vrf" configurations are sent to the device. If you change the setting to "true", "alias ip-vrf" configurations are not sent to the device.

Setting the "alias exec" Command

The IpsaConfigVersion alias command **alias exec IpsaConfigVersion** sent at the end of every configuration session can be disabled or enabled by using the property **IpsaConfigVersionAliasDisabled** in the **default.properties** file. The default.properties file is located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

- true: Disables the generation of the alias by IP Service Activator.
- false (default): Enables the generation of the alias.

This property is available as a check box in the ConfigGUI tool as a Configurable Parameter under Network Processor Framework -> Common -> Ipsa Config Aliases disabled.

Enabling the Cisco cartridge precheck **preCheckConfigVersion** is redundant and has no effect when the alias is disabled. The **networkprocessor.log** is generated with a warning message describing the redundancy as *"precheck-cisco-config-version is redundant when ipsa-Config-Version-Alias-Disabled is set true"*.

Synchronizing the versions when aliases are in conflict and when alias is disabled is not possible and generates the following warning message in the UI: "*Not possible to synchronize versions when IPSA Config Version is disabled*".

Note: Changing the property setting from **false** to **true**, or updating the patch with this change on top of the existing IP Service Activator installation, and with the property set to Changing the property setting from false to true, or updating the patch with this change on top of the existing IP Service Activator installation, and with the property set to true, removes all of the alias exec IpsaConfigVersion commands from the devices connected to that particular network processor instance to ensure that the aliases generated previously are removed according to the property setting., removes all of the alias **exec IpsaConfigVersion** commands from the devices connected to that particular network processor instance to ensure that the aliases generated previously are removed according to the property setting.

Adding the "unmanaged RIP redistribution" Command for Juniper VPNs

You can add the property **junosUnmanagedRIPRedistribution** to the **default.properties** file to control whether RIP will be added to the protocol list in both import and export policy-statements.

The **default.properties** file is located in the following directory:

/opt/ServiceActivator/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor

If you set the **junosUnmanagedRIPRedistribution** property to "false", RIP is not added to the protocol list in import and export policy-statements. If you change the setting to "true", RIP is automatically added to the protocol list in both import and export policy-statements.

For example, if you add the property and set it to "true", the line in the **default.properties** file appears as:

junosUnmanagedRIPRedistribution = true

Configuring the Network Processor to Accept Exact Names on a Huawei Device

Huawei devices restrict the length of some QOS-related names. You can configure the Network Processor to accept QOS names exactly as you enter them. For information about changing this property, see *IP Service Activator Huawei Cartridge Guide*.

Command Delivery Modes

In addition to regular **Online mode**, in which configuration commands are sent to devices, IP Service Activator supports two offline modes of operation: **Offline Test mode** and **Offline Maintenance mode**. The variety of Offline mode options is summarized in Table 7–1.

Table 7–1 Scope of the Offline Modes

Scope of the Offline mode	User interface	Offline modes available	For more information, refer to
device level (affects all interfaces and sub-interfaces on a cartridge-driven device)	GUI	Offline Maintenance Offline Test	Online Help
Network Processor level (affects all cartridge-driven devices and all their interfaces/sub-interfaces)	CLUI*	Offline Maintenance (also known as FileInterface mode) Offline Test (also known as NoCommandDelivery)	"Overview of Command Delivery Mode at the Network Processor Level"

* Use command line approaches, including the ComponentParameters utility, and editing of the **cman.cfg** file.

Overview of Command Delivery Mode at the Network Processor Level

You can set the mode of operation using either the command line or the GUI. There is a functional difference between the two versions. The command-line version sets the mode for *all* cartridge-driven devices, whereas you can set the mode for each device

using the GUI version. How to use the command-line version is explained later in this document. For details about using the GUI version, refer to the offline modes topic in the Online Help.

- Use **Offline Test mode** in a production system to temporarily disable command delivery to devices. All commands are logged in the audit trail. This mode can be used, for example, during initial testing of a production system. Experienced users can also use this mode to debug problems encountered in the field. (**Offline Test mode** is sometimes called **NoCommandDelivery mode**.)
- Use **Offline Maintenance mode** in a non-production system without access to a real network. This mode allows new users to learn system behavior without affecting real devices. Offline Maintenance mode is also used to rebuild lost Device Models and to facilitate service migration into IP Service Activator. No conversion path to a fully operational system is provided. (**Offline Maintenance mode** is sometimes called **FileInterface mode**.)
- When transactions are successfully committed in Offline Maintenance mode, the target state of each device is written to a Device Model document file (one file per device). The audit trail log file shows the commands that would have been sent to devices. The Network Processor returns notifications to the Policy Server as if the commands were successfully sent to devices.
- It is not recommended to operate both modes on the Network Processor at the same time. If both modes are enabled, then Offline Test mode takes precedence.

Command Delivery Mode Behavior at the Network Processor Level

Differences in behavior between Command Delivery modes are summarized in Table 7–2.

Table 7–2 Command Delivery Mode Behavior When Set at Network Processor Level

Behavior	Offline Test mode	Offline Maintenance mode	Online mode
Sends commands to devices	No	No	Yes
Saves the Device model (see Note 1)	No	Yes	Yes
Returns success notification	No*	Yes*	Yes*
Returns failure notification (such as “Invalid data”)	Yes**	Yes**	Yes**

* Returning success notifications causes the Policy Server to change the state of concretes from Active to Installed. If not returned, concretes remain in Active state.

** Returning failure notifications changes the state of concretes to Rejected.

Note: "Saves the Device model = Yes" means that the Network Processor considers this configuration as having been successfully sent to the device, and will not resend it.

If the Device model is not saved, and the next transaction is committed while in Online mode, the Network Processor sends this configuration to the device.

If the Device model is not saved, and the next transaction is committed while in Offline Maintenance mode, the Network Processor saves the Device model and does not resend the configuration to the device.

Notifications for Network Processor

The Network Processor generates notifications. They are **TestFailed** for config failed and **TestSucceeded** for config succeeded in OfflineTest mode. The notifications are translated into concrete activation status **TestFailed** or **TestSucceeded**, for the affected concrete. This results are that the affected concretes are removed from the concrete activation status cache, which does not cause the transaction to timeout in all cases.

Setting Command Delivery Mode Using the Command Line UI

This procedure describes how to configure the Network Processor for either Offline Test mode or Offline Maintenance mode, using the command line user interface (CLUI).

Note: This action affects *all* devices assigned to the Network Processor.

Pre-requisite: The Network Processor is installed and verified.

Perform the following steps on the host where the Network Processor is installed:

1. Edit the **cman.cfg** file (in directory **/opt/OracleCommunications/ServiceActivator/Config**) to add the desired mode option in the Network Processor line. (Removing the option restores all devices to **Online mode**.)

   To configure Offline Maintenance mode, Add the **FileInterface** option.

   An example follows:

   ```
   #NetworkProcessorEntry
   /opt/OracleCommunications/ServiceActivator/bin/networkprocessor
   "-FileInterface -ComponentName proxy-np-srvotlab481
   -ComponentLocation srvotlab481" 1 60 1 0
   #End
   ```

   To configure Offline Test mode, add the **NoCommandDelivery** option.

   An example follows:

   ```
   #NetworkProcessorEntry
   /opt/OracleCommunications/ServiceActivator/bin/networkprocessor
   "-NoCommandDelivery -ComponentName proxy-np-srvotlab481
   -ComponentLocation srvotlab481" 1 60 1 0
   #End
   ```

Notes:

If switching from Offline Test mode to Online mode, any pending configuration is sent to devices upon Network Processor restart.

If both Command Delivery mode options are configured, Offline Test mode takes precedence. It is not recommended to operate both modes on the Network Processor at the same time.

2. Stop and restart the Component Manager:

```
$ ipsacm stop
$ ipsacm start
```

Component Manager restart turns on the configured Command Delivery mode for *all* devices assigned to the Network Processor.

Audit Trail - Indication of Operational Mode

The following sample Audit Trail log file entries indicate the date and time, IP address of the device, the mode of operation, and the command sent. (No indication of mode is given in the Online mode.)

Offline Test (NoCommandDelivery) Mode:

```
2005-03-28 16:51:15,008|10.13.4.11|no-command-delivery|save
```

Offline Maintenance (FileInterface) Mode:

```
2005-03-28 16:55:22,754|10.13.4.11|file-interface|interface Atm3/0.32
```

You can edit the parameter **useOfflineCommitOptions** in the network processor **default.properties** file so that the audit trail shows the offline mode being used.

The **default.properties** file is in the directory:

Service_Activator_Home`/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor.`

The default value of **useOfflineCommitOptions** is **false**, meaning that the audit trail will be logged as usual, showing either “no-command-delivery” or “file-interface” is being used.

If you set the **useOfflineCommitOptions** value to **true**, the network processor will log the audit trail with “offline-maintenance” for Offline Maintenance mode and “offline-test” for Offline Test mode.

The following section of the file **default.properties** shows the offline modes:

```
# When set to 'true', change the command delivery mode display string in the audit
log from "file-interface" and "no-command-delivery" to "offline-maintenance" and
"offline-test", respectively.
useOfflineCommitOptions = false
```

Switching Between Offline Test Mode and Online Mode

There are two methods for switching between Offline Test mode and Online mode at the Network Processor level.

Method 1

Use the ComponentParameters interface to change the mode of the Network Processor while it is running. (For information about using ComponentParameters, see "Generating debug log files for non-Java based IP Service Activator components".) This change is recognized only until the next restart of the Network Processor. To make the change permanent, use Method 2.

Method 2

Perform the procedure "Setting Command Delivery Mode Using the Command Line UI".

Logging with the Network Processor

For details about Network Processor logging, see "Network Processor Logging".

Network Processor Database-related Log Files

For details about database-related log files see "Network Processor Database-related Log Files".

Synchronization and Recovery Strategies and Tasks

This section describes utilities and procedures for detecting and correcting the loss of synchronization between the IP Service Activator Last Device State (LDS) and the actual device configuration.

Loss of synchronization means that the configuration version timestamp of the LDS is different than the configuration version timestamp on the device. Loss of synchronization can be detected during an audit of a device. Detecting a loss of synchronization can raise a critical fault against a device, putting it into Intervention Required state. (The severity of this fault is configurable. For details, see "Setting the Severity of the Loss of Synchronization Fault".)

The LDS and the device are re-synchronized when their timestamps are made the same. Re-synchronization is performed after any other configuration differences are corrected. Tools you can use to correct configuration differences include Audit, Audit Synonyms, Command Re-issue, and Migration.

Causes of Configuration Mismatches

Ideally, the LDS always matches the actual device configuration. In reality, configuration mismatches can arise in several ways:

- Configuration is sent to the device manually instead of through IP Service Activator.
- Configuration is sent while the device is in Offline Maintenance mode, which updates the LDS, but not the device itself.
- An audit returns apparent configuration differences, reflecting reformatting by the device of the configuration sent by IP Service Activator.

Configuration Mismatch Recovery Scenarios

Introduction

A version timestamp in the device model and on the device ensures that IP Service Activator and the device have the same view of the device configuration. If the version timestamp in the last device state and on the device itself do not match, then a fault is raised against the device object. In some circumstances the fault raised is critical, putting the device into Intervention Required state. All new provisioning on the device should stop until the IP Service Activator view of the device is synchronized with the actual device.

Three synchronization recovery scenarios are described:

- The device configuration is older than the last device state.
- The device configuration is newer than the last device state.
- The device timestamp and last device state timestamp match, but the audit detects differences.

Device Configuration Is Older than the Last Device State

Condition: The configuration timestamp on the device is older than the timestamp in the last device state, or, there is no configuration timestamp on the device and a configuration timestamp exists in the last device state.

Potential Causes of this Mismatch

- The device configuration was restored from an older version.
- The device configuration was deleted and no backup is available.
- The device configuration timestamp was removed or changed.

How this Version Mismatch is Detected

- A fault is raised on the device because the audit has detected a timestamp difference between the device and the last device state.
- A fault is raised on the device because a pre-check has detected a timestamp difference between the device and the last device state.
- The Audit screen shows the configuration timestamp from the last device state and the device configuration timestamp. The user sees a timestamp difference between the two.

Device Configuration Is Newer than the Last Device State

Condition: The configuration timestamp on the device is newer than the timestamp in the last device state, or, there is no last device state, or, the last device state does not contain a configuration timestamp while a configuration timestamp does exist on the device.

Potential Causes of this Mismatch

- The IP Service Activator database was restored from an older version.
- The Network Processor failed before the LDS was updated, but after the configuration was successfully sent to the device.
- The Database connection failed or the write action to the database failed after the configuration was successfully sent to the device.

- The device returned an error after the configuration timestamp was successfully sent to the device, then command rollback failed. This cause would be rare but possible.

How this Version Mismatch is Detected

- A fault is raised on the device because the audit has detected a timestamp difference between the device and the last device state.
- A fault is raised on the device because a pre-check has detected a timestamp difference between the device and the last device state.
- The Audit screen shows the configuration timestamp from the last device state and the device configuration timestamp. The user sees a timestamp difference between the two.

Device and Last Device State Timestamps Match but Audit Detects Differences

Condition: The configuration timestamp on the device matches the one persisted in the last device state but configuration differences are detected in the audit.

Potential Causes of this Mismatch

- The device configuration has been manually changed.
- Services have been created, changed, or deleted in IP Service Activator while the Network Processor, Device, Interface, or Sub-Interface were in Offline Maintenance Mode. The changes were persisted to the last device state but not sent to the device.
- Audit differences can be caused if the device reformats configuration on the device so that it no longer appears as expected by IP Service Activator. (This can usually be fixed by using the Audit Synonyms feature of the cartridge.)

How this Version Mismatch is Detected

- An audit must be performed to detect these differences.

Configuration Mismatch Recovery

Pre-requisite

Understand the impact of operating in Offline Maintenance mode, and take the necessary action to stop all provisioning against the device being re-synchronized.

Impact of Operating in Offline Maintenance Mode

During the process of re-synchronizing the device model with the device, it may be necessary to take the device out of Intervention Required state and put it into Offline Maintenance mode. This is necessary so that device information can be sent to the Network Processor to match the device model view with the device.

If an external system is sending orders to IP Service Activator while the device is being re-synchronized, it may send orders to IP Service Activator for this device. When the device is in Offline Maintenance mode, any outstanding or new orders will be inserted into the device model without being sent to the device, making the device model *more* out of synch with the device.

It is highly recommended to stop regular provisioning on a device while it is being synchronized. If you do not, then all orders processed while the device was in Offline

Maintenance mode will need to be re-synchronized with the device before you put the device into Online mode.

Recovering from Configuration Mismatch Conditions

1. Switch the device to Offline Test mode.
2. Display the **Device properties - Audit/Migrate** page in the user interface. Select the **Help** button to open the Online Help file for this page. Read the help text to understand the impact of using the commands in this property page.
3. On the **Audit** tab, select **Initiate** to start an audit of the device, selecting the **Show all device commands** option.

   The Audit response pane displays the results of the audit.
4. For every command in RED ('missing from the device'), decide if the command(s) should be on the device.

   Note: If the device configuration is older than the last device state, the missing configuration could be services that were added after the device configuration was backed up. If the device configuration is newer than the last device state, this missing configuration could be services that were deleted after the IP Service Activator database was backed up.

5. If the missing configuration *should be* on the device, understand these impact statements, then perform the following actions.

   Possible impacts of reissuing commands: If there is already a command on the device similar to the command being re-issued, one of four different behaviors can occur with the **Re-issue Commands** action:

   - Re-issue Commands replace the existing command on the device.
   - Re-issue Commands add the new command but the old command remains on the device as well. The old version of the command shows as blue (conflict) in a a subsequent audit. In this case, manually remove the old command from the device.
   - Re-issue Commands add the new command but it may not be in the right sequence in relation to other commands on the device. In this case, manually re-order the command sequence on the device.
   - Re-issue Commands fail because the old version of the command must be removed before the new one can be added. In this case, manually remove the old version of the command before re-issuing the command.
   - Highlight the commands to be reissued and click **Re-issue Commands** on the **Command Re-issue** tab.
   - Monitor the **Faults** pane in the GUI to ensure that the commands are successfully sent.

     An Info-level fault appears when command re-issue completes successfully; otherwise an Error-level fault appears.
   - Run the audit again, to ensure that RED commands now show in BLACK, and that no BLUE commands are showing.
6. If the missing configuration *should not be* on the device, locate the policy (in the IP Service Activator GUI) that is generating the configuration.

- Switch the device to Offline Maintenance mode.
- If the whole configuration that the policy is generating is no longer required, disable its concrete.
- If only some of configuration is no longer required, modify the policy to match the desired configuration.
- If a critical fault exists against the device, clear it to move the device out of the Intervention Required state.
- Commit your changes.
- If the policy was modified, wait for its concretes to go to the Installed state. If the policy concrete was disabled, wait for its concretes to go to the Uninstalled state.
- **Initiate** the **Audit** for the device again, and ensure there are no other missing commands.
- If the policy concrete was disabled, remove the concrete by the appropriate means. (For example: delete the policy, unlink the policy, or change or remove a role.)

7. For every command in BLUE (conflicting or manual configuration), decide if this command should be on the device.

> **Note:** If the device configuration is older than the last device state, the missing configuration could be services that were deleted after the device configuration was backed up. If the device configuration is newer than the last device state, this missing configuration could be services that were added after the IP Service Activator database was backed up.

8. If this configuration *should not be* on the device:

   Connect to the device and manually remove the configuration.

   Run the audit again and ensure that no conflict commands (BLUE text) appear.

9. If this configuration *should be* on the device, decide if this configuration should be managed by IP Service Activator.

   - If this configuration *should not be* managed by IP Service Activator, do nothing further for this command.
   - If this configuration *should be* managed by IP Service Activator, create or modify policies in the IP Service Activator GUI to properly represent the conflicted configuration.
     - Switch the device to Offline Maintenance mode. For a description of the impact of doing this, see "Configuration Mismatch Recovery".
     - If a critical fault exists against the device, clear it to move the device out of the Intervention Required state.
     - Commit your changes.
     - Run the audit for the device again, and ensure there are no other conflicted commands.
     - If differences are detected because auto-generated names or ids are different:

On the **Migrate** tab, run **Migration** with **Preview Only** selected, to ensure any generated configuration names and ids are migrated to the last device state.

If Migration with Preview Only was successful, then deselect **Preview Only** and **Initiate** the **Migration** again.

Initiate the **Audit** for the device again, and ensure there are no other conflicted commands.

10. If there are no remaining differences in the Audit response, go to step 11.

    If only the configuration timestamp difference remains in the audit response, continue at step 10.

11. Click the **Synchronize** button on the **Synchronize Versions** tab.

    The configuration timestamps are re-synchronized, so that a fault is not raised against the device when an audit is performed.

    An Info-level fault indicates that the synchronize action was successful.

> **Note:** Since the configuration version timestamp is an important indicator of device configuration changes, it should not be the only command sent to the device. In this step, you have updated the configuration version timestamp in the LDS and on the device only after carefully re-establishing the integrity of the device configuration.

12. If the device is in Offline Maintenance mode, change its command delivery mode to **Offline Test**.
13. **Commit** the changes.
14. Run the audit again and ensure that the audit is completely successful.
15. Change the command delivery mode to **Online**.
16. If a critical fault **configuration timestamp is mismatched** still exists against the device, delete the critical fault.
17. Commit the changes.

    Regular provisioning can resume.

Setting the Severity of the Loss of Synchronization Fault

The loss of synchronization fault (message 3308) is raised when a configuration version timestamp mismatch is detected between the last device state and the device.

You can modify the severity level of this fault message to be one of six values: **Critical**, **Error**, **Warning**, **Notice**, **Info**, **Suppress**.

To do this, edit the parameter **lossOfSyncSeverity** in the network processor **default.properties** file, located in *Service_Activator_Home*/**Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor**.

Syntax:

```
lossOfSyncSeverity = Critical
```

Notes:

When the message is raised with severity **Critical**, the device is put into **Intervention Required** state. In this state, the device no longer accepts configuration commands.

If the parameter is omitted from the **default.properties** file, the default setting **Critical** is applied to this fault.

When the value **Suppress** is configured, no fault is raised.

Dropping Database Tables/User

In earlier versions (from 5.2.2 to 5.2.3) of database tables for IP Service Activator user, if the tables were dropped, the data related to Network Processor was not removed from the database. For 5.2.4 or later, it is recommended to drop the user as this cascades down and removes every table under it. Dropping the user removes all the objects in the IP Service Activator database schema.

In the IP Service Activator database, apart from table objects, there are sequences, views, packages, functions, and types (also indexes, but they do not get dropped when you drop tables). If these other objects are not dropped, the database schema becomes corrupted. Do the following:

- Drop the user using the following command.

  ```
  DROP USER user CASCADE
  ```

  Warning: This command removes all the objects in the IP Service Activator database schema.

Moving a Device to Another Network Processor

This procedure describes how to move devices off one Network Processor onto another Network Processor. This action may be warranted under several scenarios:

- Recovery scenario: If the server on which the Network Processor is installed fails, you can recover operations if you have a second Network Processor on another server.
- Load-balancing scenario: You may want to move one or more devices from one Network Processor to another for load-balancing purposes.

Ensure that your capabilities, options, synonyms, registry, and other settings are in synchronization between the source and target network processors, or you could strip configuration from the target device.

When you move a device from one Network Processor to another, the new Network Processor reads the Last Device State for the device from the database. Each LDS stores the vendor cartridge type as a key, so the correct LDS is retrieved by the new Network Processor.

To move a device to another Network Processor by using the GUI:

1. On the **Topology** tab, right-click a device that you want to move.

   To move more than one device, select multiple devices.

2. In the **Device** dialog box, **Management** property page, select a different network processor in the **Proxy Assignment** list.

3. Click **OK**.

   When the Device dialog box appears, it applies your action to all selected devices. A warning message appears indicating multiple objects will be edited.

4. Click **OK**.

   A warning message appears indicating multiple objects will be edited.

5. Click **OK**.

Cartridge capabilities management

Capabilities in IP Service Activator are used to manage the type of configuration that can be assigned to a device and its interfaces. They can be used to restrict configurations regardless of what the device or interface is actually capable. This is useful for organizations that want to restrict the types of configurations that they want to manage with IP Service Activator.

Capabilities are used for validation in the Policy Server, if the service does not validate against the device or interface capabilities, no associated concrete is created for the service and a warning is raised in the IP Service Activator fault panel.

Capabilities do not affect the style of the generated configuration. The default IP Service Activator capabilities configuration enables capabilities for all the services. The network processor installation provides a tool for generating new default capabilities files for the devices. These files must be edited to further restrict what is supported by each IOS, device and interface combination. For more information, refer to Cisco Cartridge Configuration Utility in *IP Service Activator Cisco IOS Cartridge Guide*.

Capabilities and the Registry

The association of a specific capabilities configuration is registered in the Network Processor Cartridge registry file for each specific vendor device type and OS version. Here is an example of the registry format used for capabilities assignment for a device type and IOS.

```
<cartridgeUnit>
<name>com.metasolv.serviceactivator.cartridges.cisco.units.cu1.7500 </name>
  <driverType>cisco</driverType>
  <deviceType>Cisco 7507</deviceType>
  <osVersion>12.0</osVersion>

<smToDmQuery>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/sm2dm.xq</sm
ToDmQuery>

<dmValidation>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/dmValidatio
n.xq</dmValidation>

<dmMigration>com/metasolv/serviceactivator/cartridges/cisco/xquerylib/dmMigration.
xq</dmMigration>

<dmToCliQuery>com/metasolv/serviceactivator/cartridges/cisco/units/cu1/annotatedDm
2Cli.xq</dmToCliQuery>
  <capabilities>com/metasolv/serviceactivator/cartridges/cisco/capabilities/cisco_
7500.xml</capabilities>
  <options>com/metasolv/serviceactivator/cartridges/cisco/options/cisco_7500_
12.0.xml</options>

<errorMessages>com/metasolv/serviceactivator/cartridges/cisco/messages/errorMessag
```

```
es.xml</errorMessages>

<warningMessages>com/metasolv/serviceactivator/cartridges/cisco/messages/warningMe
ssages.xml</warningMessages>

<successMessages>com/metasolv/serviceactivator/cartridges/cisco/messages/successMe
ssages.xml</successMessages>
</cartridgeUnit>
```

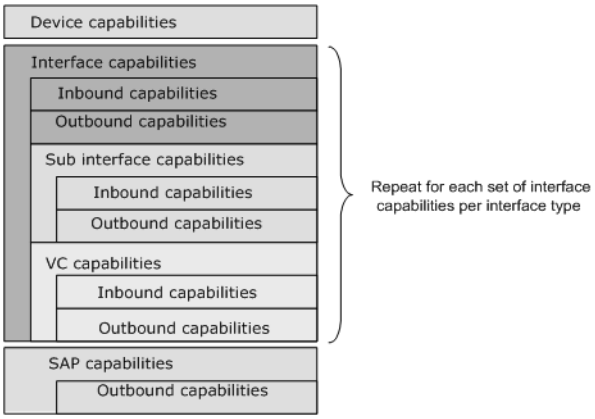
For more information, refer to Cisco Cartridge Configuration Utility in *IP Service Activator Cisco IOS Cartridge Guide*.

Capabilities File Structure

Each capabilities file defines a single set of device level capabilities, multiple groups of interface capabilities, and optionally a set of Service Application Points (SAP) capabilities.

Figure 7–4 describes the structure of the capabilities XML files generally take.

Figure 7–4 Example of the XML Structure of the Capabilities Files



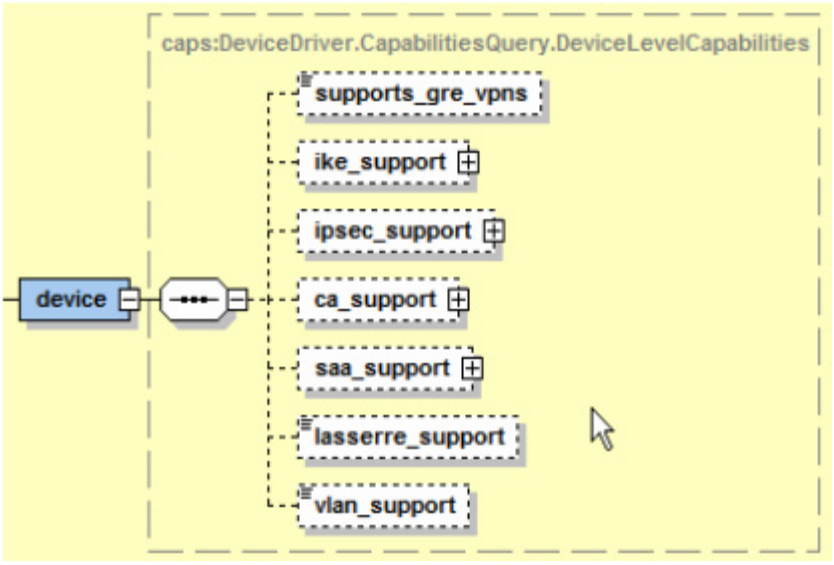
Device Capabilities

Capabilities defined at the device level and are used to control what protocols and services a particular device is capable of supporting. It is important to note that even though a device may have the ability to support a feature set, the feature set may not be supported by the IOS currently installed on the device.

Figure 7–5 shows an example of device level capabilities for SAA support under a Cisco device.

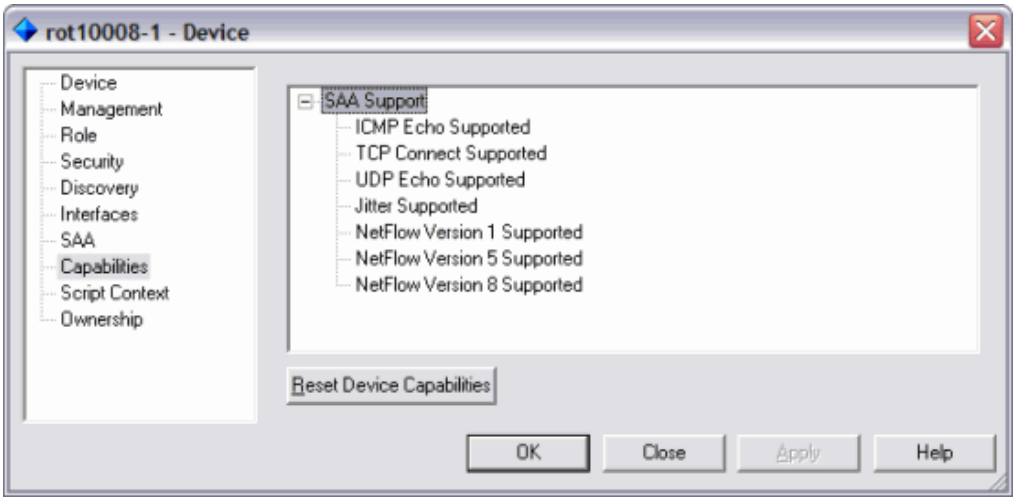
```
<caps:device>
   <caps:saa_support>
      <caps:operations_supported>1</caps:operations_supported>
      <caps:saa_types_support>15</caps:saa_types_support>
      <caps:netflow_version_support>7</caps:netflow_version_support>
   </caps:saa_support>
</caps:device>
```

Figure 7–5 Example of Device Level Capabilities



At present, the supported capability types are SAA, Lasserre (TLS), and VLAN. To view the capabilities currently assigned to a device, right-click the device and select **Properties**, and select **Capabilities** from the side menu, as shown in Figure 7–6.

Figure 7–6 Device Capability Dialog Box

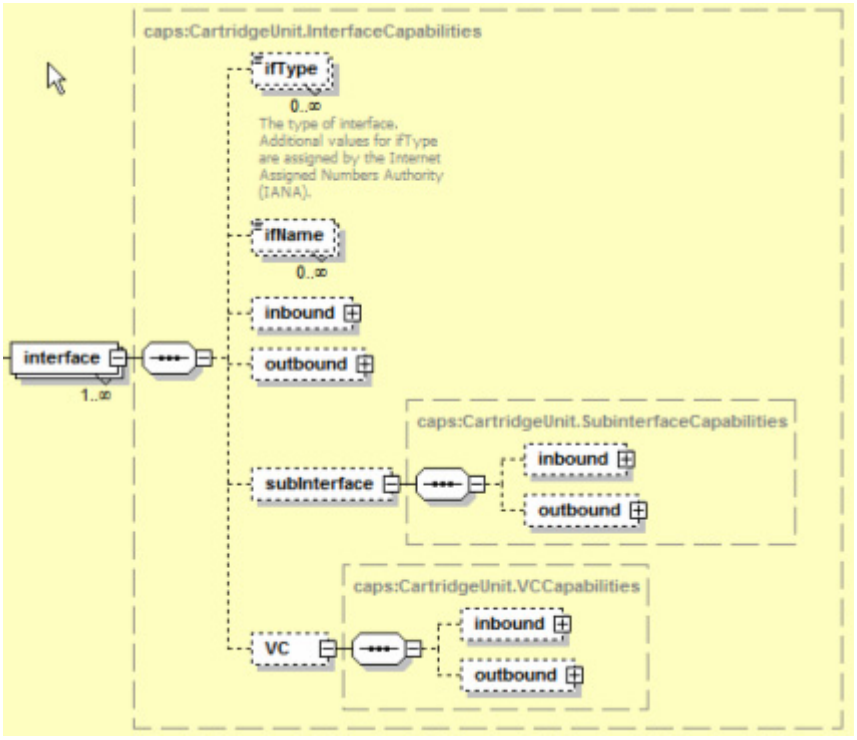


See the device capabilities reference table in "Capabilities classifications" for details on each device level capability.

Interface Capabilities

Interface inbound and outbound capabilities are defined for each main level interface, any sub-interfaces (optional), and any layer 2 VC capabilities (optional). Interface level capabilities do not provide further granularity to the supported device level capabilities. There is no relationship between device and interface level capabilities. Additionally, capabilities assigned to an interface bear no relationship to and do not restrict capabilities assigned to sub-interfaces and layer 2 interfaces. Figure 7–7 displays a visible example of these relationships.

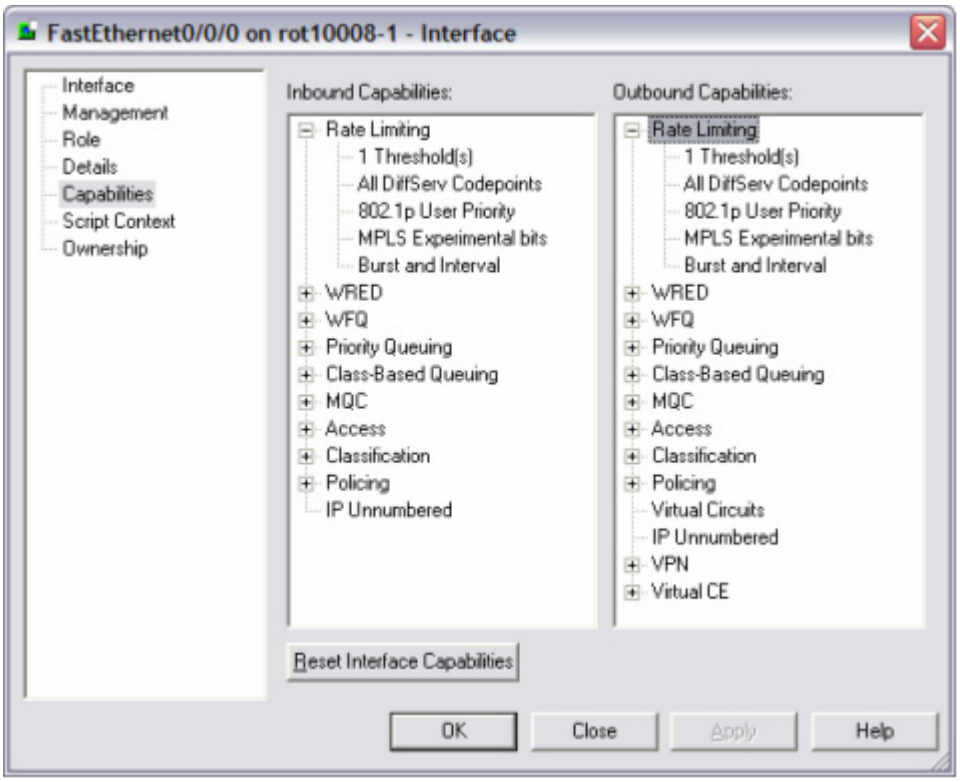
Figure 7–7 Example of Interface Level Capabilities



To view the capabilities currently assigned to an interface right-click the interface, select **Properties**, and select **Capabilities** from the side menu.

Figure 7–8 displays the interface capability dialog box.

Figure 7–8 Interface Capability Dialogue Box



Multiple interface types can share the same interface capabilities definition by declaring multiple **ifType** elements. The **ifType** element is a standard SNMP MIB interface type as defined by the IANA. For more information on IANA ifType definitions please refer to the following URL: http://www.iana.org/assignments/ianaiftype-mib.

> **Note:** If no ifType is defined for a capabilities definition it is used as the default catch-all capability. Interface types that are discovered which are not currently referenced in the capabilities file receives this default capability. If no default capability is defined that will not receive capabilities.

```
<caps:capabilities>
…
<caps:interface>
<caps:ifType>17</caps:ifType><!--sdlc -->
<caps:ifType>32</caps:ifType><!--FrameRelay -->
<caps:ifType>77</caps:ifType><!--lapd-->
<caps:ifType>171</caps:ifType><!-- POS -->
<caps:inbound>
…
</caps:inbound>
<caps:outbound>
…
</caps:outbound>
…
</caps:interface>
```

SAP Capabilities

Service Application Points are used to model services that are not applied explicitly to an interface. SAP capabilities have a separate section of a capabilities definition that is not tied to the interface or device definitions. SAPs can be modeled and linked to a site and are used to represent configuration that resides on a device but is not explicitly assigned to an interface. When SAP capabilities are defined for a device/IOS combination, they restrict the type of configuration a user may apply to new or existing SAPs.

Inbound and Outbound Capabilities

Each set of inbound and outbound capabilities definitions follow the same general structure for defining the inbound or outbound capabilities supported for the specific interface, sub-interface, VC, or SAP context.

Capabilities and Options

Capabilities define the types of configurations hardware will support. Options however, are used to further refine how configuration is supported on a hardware and IOS combination.

Vendors with large numbers of hardware and software variants have allowed for non-standard configuration commands to exist. Options are a way of explicitly supporting a particular and often obscure command variant for a device or IOS. Additionally, you can create options to support IOS implementations that contain bugs or have unexpected behavior.

Additionally, there is often more than one way to apply a particular configuration to a router. Options allow the provider to implement a preferred method or standard of configuration.

For more information, refer to Options Framework in *IP Service Activator Cisco IOS Cartridge Guide*.

Default Capabilities Generation

When you generate capabilities for a device, by default, all capabilities are open. If a configuration is provisioned that is not supported by a device/IOS/interface combination, the commands will fail for the device. The capabilities file must be manually edited so that only the supported or desired features can be provisioned and sent to device.

Initial Capabilities Configuration

Consider the following factors when determining the initial capabilities definition for a particular device:

- The current routing configuration.
- What configuration types the hardware supports.
- What type of configuration you want to perform.
- The type of configuration the IP Service Activator cartridge currently supports.
- Known issues or bugs with ISO versions in use, that may limit or constrain allowable configurations.

Modifying and Creating Capabilities Files

To modify or create the capabilities files:

1. Modify the capabilities file (or create a new capabilities file to modify).
2. If it is a new capabilities file, update the registry to reference the appropriate capabilities file. Reload the registry using the **npAdmin.sh** tool.
3. Un-manage the device.
4. Right-click the device, select **Properties** > **Capabilities** > **Reset Device Capabilities** and commit. This resets the capabilities of the device or interface.
5. Right-click the device and select **Discover** (or, **Discovery** > **Fetch Capabilities**; for the whole subnet).
6. Re-manage the device.

If an update is limited to the capabilities file, you do not need to reload the registry.

Enabling, Disabling, or Modifying a Capabilities Definition

There are three ways of disabling capabilities in a capabilities file. You can disable capabilities by commenting out the element in the capabilities file using an XML comment. Alternatively, you can delete a specific capability element from the document. Finally, many capabilities can be disabled by setting their value from 1 to 0. You can also enable capabilities by setting their value from 0 to 1. It is important to ensure that when a capabilities file is edited that it still conforms to the capabilities schema and follows correct XML formatting.

Here is an example of an XML comment:

```
<caps:device>
<!-- <caps:saa_support>
<caps:operations_supported>1</caps:operations_supported>
<caps:saa_types_support>15</caps:saa_types_support>
<caps:netflow_version_support>7</caps:netflow_version_support>
</caps:saa_support> -->
</caps:device>
```

Here is an example of enabling/disabling priority queuing - pq:

```
<caps:pq>
<caps:support>0</caps:support>
<caps:classification>
<caps:supported>SRC_IP</caps:supported>
<caps:supported>DST_IP</caps:supported>
<caps:supported>SRC_PORT</caps:supported>
<caps:supported>DST_PORT</caps:supported>
<caps:supported>DIFFSERV</caps:supported>
<caps:supported>IPv4PRECEDENCE</caps:supported>
<caps:supported>IPv4TOS</caps:supported>
<caps:supported>URL</caps:supported>
<caps:supported>MIME</caps:supported>
<caps:supported>APP_PROTO</caps:supported>
<caps:supported>APP_NAME</caps:supported>
<caps:supported>DOMAIN_NAME</caps:supported>
<caps:supported>IEEE_802_1_PBITS</caps:supported>
<caps:supported>MPLS_EXP</caps:supported>
<caps:supported>MPLS_TOPMOST</caps:supported>
<caps:supported>CLASS_MAP</caps:supported>
<caps:supported>MATCH_ANY</caps:supported>
```

```
<caps:supported>TCP_HEADER_OPTIONS</caps:supported>
<caps:supported>TCP_ESTABLISHED</caps:supported>
<caps:supported>EXCLUDE_CLASSIFICATION</caps:supported>
<caps:supported>ICMP_HEADER_OPTIONS</caps:supported>
</caps:classification>
</caps:pq>
```

Adding a New Interface Type to an Interface Capabilities Definition

Once you define a capability set for a particular interface type, you can add new interfaces to the set. To do this, add the new interface type number to the appropriate caps definition in the Capabilities.xml file. For more information, refer to "Re-initializing Capabilities for a Device or Interface" and "Interface Capabilities".

Modifying Capabilities for a Specific Vendor Device Type or OS Version

Another common scenario is to create restricted capabilities for a specific device type or OS version to prevent the user from attempting to provision service configurations that the device does not support. To perform this restriction, create a new capabilities file by copying the existing open capabilities file and rename it. The name should represent the device type of IOS the file you intend to restrict. Edit the file as necessary and reference the file in the **Mipsa_Registry.xml** file. Restart the network processor to initialize the new set of capabilities. The capabilities sometimes affect a previously discovered device. For more information about how to deal with this situation, refer to "Re-initializing Capabilities for a Device or Interface".

Capabilities Dialogue Messages

These messages appear in the capabilities dialogue when a fetch does not occur or it is not successful. These are the two messages that you may see:

- **No specific capabilities reported**: implies that there is no capability defined in the capabilities file for the device or interface. The device is discovered but the network processor is unable to match a specific capability to the hardware.
- **Capabilities not obtained**: implies that the device of an unknown type is discovered. No device representing the device type/IOS combination is defined in the **MIPSA_Registry.xml** file. This can also occur if a device is unreachable or the wrong username or password is defined for the device during discovery.

Re-initializing Capabilities for a Device or Interface

To reset the capabilities for a device or interface:

1. Right-click the device, select **Unmanage** and commit.
2. Right-click the device and select **Properties**.
3. Select **Capabilities** in the **Properties** dialogue
4. Click **Reset Device/Interface Capabilities** in the **Capabilities** dialog and commit.
5. Rediscover the device to apply the new capabilities set to the device and interfaces.

> **Note:** Warning messages should be checked to ensure that the new capabilities match the current configuration or when a device is re-managed configuration may be stripped from the router.

Assigning unsupported capabilities to a device or cartridge

Not all devices support all the capabilities. In this case, if you push a configuration to a device that the device does not support; the device rejects it and causes a configuration rollback. If a particular a cartridge does not support a capability, and error message appears indicating that the configuration sent to the router resulted in no commands.

Note: You must stop the network processor and restart it to pick up any capabilities file changes.

Capabilities reference

This section details the capabilities presently supported by IP Service Activator. Here are the definitions of the parameters used in the tables below:

- Boolean: of 0 or 1. The Value 1 indicates that the capability is supported.
- Integer: any whole number. When set to 0, indicates the capability is not supported.
- Additive: an integer value representing a bitmap. Values are added (bitwise AND operation) together to arrive at a supported set. For example, in the set

```
1 = UBR
2 = CBT
4 = RTVBR
8 = NRTVBR
16 = ABR
```

enabling UBR and CBT would mean setting a value of 3.

Device capabilities

Table 7–3 lists the supported device capability element, and type.

Table 7–3 Device Capability Element Definitions

Element	Type/Description	Supported
<caps:ca_support>	Boolean	Yes
<caps:supports_ca>	Boolean	Yes
<caps:supports_ca_trusted_roots>	Boolean	Yes
<caps:saa_support>	Boolean Container for Service Assurance Agent capabilities supported by the device.	Yes
<caps:operations_supported>	Boolean Service Assurance Agent operations (probes) supported.	Yes
<caps:saa_types_supported>	Integer Supported SAA probe types. Bitfield addition of: 1 = ICMP Echo 2 = TCP Connect 4 = UDP Echo 8 = Jitter	Yes

Table 7–3 (Cont.) Device Capability Element Definitions

Element	Type/Description	Supported
<caps:netflow_version_support>	Integer Supported Netflow versions. Bitfield addition of 1 = Netflow V1 2 = Netflow V5 4 = Netflow V8	Yes
<caps:lasserre_support>	Boolean Lasserre (TLS or VPLS) supported by the device.	Yes
<caps:vlan_support>	Boolean Virtual LAN supported by the device.	Yes

Interface capabilities

Table 7–4 lists and describes the interface capability element, and type.

Table 7–4 Interface Capability Element Definitions

Element	Type/Description
<caps:qos_mechanism_support>	Container for QoS capabilities supported.
<caps:rate_limiting>	Capabilities supported for Rate Limiting.
<caps:supports_burst>	Boolean Rate Limiting supports Burst Rate and Burst Interval. 1=on, 0=off.
<caps:rate_limit_support>	Container for Rate Limit supported.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:wred>	Capabilities supported for Weighted Random Early Detection.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:wred_ecn>	Boolean Weighted Random Early Detection with Explicit Congestion Notification supported.
<caps:wfq>	Capabilities supported for Weighted Fair Queuing.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:supports_high_priority_percent>	Boolean Supports High Priority Percent.
<caps:supports_low_priority_percent>	Boolean Supports Low Priority Percent.
<caps:wred_support>	Container for Weighted Random Early Detection supported.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:wfq_support>	Container for Weighted Fair Queue supported.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:supports_fr_de>	Boolean Supports Frame Relay Discard Eligibility.
<caps:wrr>	Capabilities supported for Weighted Round Robin.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:pq>	Capabilities supported for Priority Queuing.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:cbq>	Capabilities supported for Class Based Queuing.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:frts>	Capabilities supported for Frame Relay Traffic Shaping.
<caps:supported>	Boolean Frame Relay Traffic Shaping supported.
<caps:becn_adapt>	Boolean Backward Explicit Congestion Notification supported.
<caps:fecn_adapt>	Boolean Forward Explicit Congestion Notification supported.
<caps:frf12>	Boolean Frame Relay Fragmentation 12 supported.
<caps:atmqos>	Integer Supported ATM traffic types. Bitfield addition of: 1 = UBR 2 = CBT 4 = RTVBR 8 = NRTVBR 16 = ABR
<caps:mqc>	Capabilities supported for Modular QoS CLI.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:llq>	Boolean Low Latency Queuing supported.
<caps:llq_bandwidth_type>	Integer LLQ bandwidth types supported. Bitfield addition of: 1 = Absolute 2 = Percentage 4 = Remaining Percent 8 = Default
<caps:cbwfq>	Boolean Class Based Weighted Fair Queuing supported.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:cbwfq_bandwidth_type>	Integer CBWFQ bandwidth types supported. Bitfield addition of: 1 = Absolute 2 = Percentage 4 = Remaining Percent 8 = Default
<caps:cbwfq_flow_queue_limit>	Boolean Class Based Weighted Fair Queuing supports queuing limits.
<caps:queue_limit>	Boolean Queue Limit supported.
<caps:single_rate_police>	Boolean Single Rate Policing supported.
<caps:policing_rate_type>	Integer Policing bandwidth types supported. Bitfield addition of: 1 = Absolute 2 = Percentage 4 = Remaining Percent 8 = Default
<caps:single_rate_police_action>	Container for Single Rate Policing supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:aggregate_policer>	Boolean Aggregate Policing supported.
<caps:two_rate_police>	Boolean Two Rate Policing supported.
<caps:two_rate_police_action>	Container for Two Rate Policing supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:shape>	Boolean Shaping supported.
<caps:shape_frts>	Boolean Frame Relay Traffic Shaping supported.
<caps:nest>	Boolean Nesting supported.
<caps:marking>	Container for Packet Marking supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:wred_support>	Container for Weighted Random Early Detection supported.
<caps:support>	Integer Number of PHBs that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:rtp_tcp_header_compression_support>	Boolean Real Time Protocol TCP Compression supported.
<caps:wred_ecn>	Boolean Weighted Random Early Detection with Explicit Congestion Notification supported.
<caps:allowed_combinations>	Container for supported QoS mechanisms.
<caps:item>	Integer Supported QoS mechanism combinations. Bitfield addition of: 1 = Rate Limit 2 = WRED 4 = WFQ 8 = WRR 16 = PQ 32 = CBQ 64 = FRTS 128 = ATM QoS Multiple <caps:item> elements can be defined to declare all supported combinations.
<caps:service_rule_support>	Container for all Service Rules supported.
<caps:rules_supported>	Integer Number of service rules that can be installed. 0=not supported.
<caps:mark_codepoints_supported>	Container for Codepoints supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:limits_supported>	Integer Number of limiting rules that can be installed. 0=not supported.
<caps:guarantees_supported>	Integer Number of guarantees that can be installed. 0=not supported.
<caps:limits_and_guarantees_supported>	Boolean
<caps:classification>	Container for Classifications supported.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:policing_rule_support>	Container for Policing Rules supported.
<caps:policing_supported>	Integer Number of policing rules that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:classification_queues_supported>	Integer Supported number of classification queues. Bitfield addition of: 1 = 1 COS queues 2 = 3 COS queues 4 = 8 COS queues
<caps:mark_codepoints_supported>	Container for Codepoints supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:access_rule_support>	Container for Access Rules supported.
<caps:access_rules_supported>	Integer Number of access rules that can be installed. 0=not supported.
<caps:classification>	Container for Classifications supported.
<caps:supported>	Traffic classification types supported. See "Capabilities classifications".
<caps:all_mechanisms_combinations>	Container for QoS mechanisms supported.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:item>	Integer Supported QoS mechanism combinations. Bitfield addition of: 1 = Rate Limit 2 = WRED 4 = WRR 8 = WFQ 16 = PQ 32 = CBQ 64 = Mark only service rules 128 = Class Queues service rules 256 = N Queues service rules 512 = Policing rules 1024 = Access rules 2048 = VPN 4096 = FRTS 8192 = ATM QoS Multiple <caps:item> elements can be defined to declare all supported combinations.
<caps:vc_support>	Boolean Virtual Circuit supported.
<caps:vpn_support>	Container for Virtual Private Networks supported.
<caps:supports_label_switching>	Boolean Supports LSP based MPLS VPNs.
<caps:supports_mpls_vpns>	Boolean Interface supports MPLS VPNs.
<caps:supports_RIP>	Boolean RIP support for PE to CE routing protocol in MPLS VPNs.
<caps:supports_static>	Boolean Static route support for PE in MPLS VPNs.
<caps:supports_OSPF>	Boolean OSPF support for PE to CE routing protocol in MPLS VPNs.
<caps:supports_eBGP>	Boolean eBGP support for PE to CE routing protocol in MPLS VPNs.
<caps:supports_EIGRP>	Boolean EIGRP support for PE to CE routing protocol in MPLS VPNs.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:supports_dhcp>	Boolean DHCP-Relay on VPN interface support.
<caps:vce_support>	Container for Virtual Circuit Endpoint supported.
<caps:supports_virtual_ce>	Boolean Interface supports Virtual CE service.
<caps:supports_RIP_virtual_ce>	Boolean RIP support for interface in Virtual CE.
<caps:supports_static_virtual_ce>	Boolean Static route creation support for interface in Virtual CE.
<caps:supports_OSPF_virtual_ce>	Boolean OSPF support for interface in Virtual CE.
<caps:supports_eBGP_virtual_ce>	Boolean eBGP support for interface in Virtual CE.
<caps:supports_EIGRP_virtual_ce>	Boolean EIGRP support for interface in Virtual CE.
<caps:point_2_point_support>	Container for Point to Point supported.
<caps:ccc>	Boolean Circuit cross connect supported (only on Juniper devices).
<caps:martini_on_main_interface>	Boolean Martini service on main interface supported.
<caps:martini_on_sub_interface>	Boolean Martini service on sub-interface supported.
<caps:encapsulation>	Integer Supported encapsulations. Bitfield addition of: 1 = PPP 2 = HDLC 4 = ATM AAL5 8 = ATM Cell 16 = Frame 32 = Ethernet 64 = Ethernet VLAN
<caps:fr_vc_creation_support>	Boolean Frame Relay VC creation supported.
<caps:ip_unnumbered_support>	Boolean IP unnumbered supported.
<caps:capability_value_pairs>	Container for Capability Value Pair supported.

Table 7–4 (Cont.) Interface Capability Element Definitions

Element	Type/Description
<caps:item>	Supported items. Multiple <caps:item> elements can be defined to declare all supported capabilities.
<caps:capability_type>	Capability type. See "Capability value pairs".
<caps:value>	Value depends on the specific capability type declared. See "Capability value pairs".

Capabilities classifications

In the following example the SRC_IP classification is used. Table 7–5 lists and describes all other capabilities classifications.

```
<caps:classification>
<caps:supported>SRC_IP</caps:supported>
<caps:supported>DST_IP</caps:supported>
</caps:classification>
```

Table 7–5 Capabilities Classifications

Classifications	Description
SRC_IP	Source IPv4 Address
SRC_IPV6	Source IPv6 Address
DST_IP	Destination IPv4 Address
DST_IPV6	Destination IPv6 Address
SRC_PORT	Source Port
DST_PORT	Destination Port
IP_PROTO	IP Protocol
DIFFSERV	Diff Serv
IPV6_DIFFSERV	IPv6 Diff Serv
IPv4PRECEDENCE	IPv4 Precedence
IPV6_PRECEDENCE	IPv6 Precedence
IPv4TOS	IPv4 TOS
URL	URL
MIME	MIME Type
APP_PROTO	Application Protocol
APP_NAME	Application Name
DOMAIN_NAME	Domain Name
IEEE_802_1_PBITS	802.1 PBits
MPLS_EXP	MPLS Experimental
MPLS_TOPMOST	MPLS Experimental Topmost
CLASS_MAP	Class Map
MATCH_ANY	Match Any
TCP_HEADER_OPTIONS	TCP Header Options

Table 7–5 (Cont.) Capabilities Classifications

Classifications	Description
TCP_ESTABLISHED	TCP Established
EXCLUDE_CLASSIFICATION	Exclude Classification
ICMP_HEADER_OPTIONS	ICMP Header Options

Capability value pairs

In the following example the MAX_RESERVED_BANDWIDTH capability is used. Table 7–6 lists and describes all of the capability value pair types.

```
<caps:item>
<caps:capability_type>MAX_RESERVED_BANDWIDTH<caps:capability_type>
<caps:value>1<caps:value>
</caps:item>
```

Table 7–6 Capability Value Pairs

Capability Types	Type/Description
MAX_RESERVED_BANDWIDTH	Boolean When used in CapabilityTypeValuePair, the value corresponding to Maximum Reserved Bandwidth is a boolean to indicate whether it is supported.
FRTS_INBOUND	Boolean When used in CapabilityTypeValuePair, the value corresponding to FRTS Inbound is a boolean to indicate whether it is supported.
LLQ_BURST_RATE	Boolean When used in CapabilityTypeValuePair, the value corresponding to LLQ Burst Rate is a boolean to indicate whether it is supported.
FRAME_RELAY_QUEUE_DEPTH	Boolean When used in CapabilityTypeValuePair, the value corresponding to Frame Relay Queue Depth is a boolean to indicate whether it is supported.
ATM_QUEUE_DEPTH	Boolean When used in CapabilityTypeValuePair, the value corresponding to ATM Queue Depth is a boolean to indicate whether it is supported.
ATM_TRANSMIT_RING_LIMIT	Boolean When used in CapabilityTypeValuePair, the value corresponding to ATM Transmit Ring Buffer Limit is a boolean to indicate whether it is supported.
SHAPING_BUFFERS	Boolean When used in CapabilityTypeValuePair, the value corresponding to Shaping Buffers is a boolean to indicate whether it is supported.

8

Managing the IP Service Activator Database

This chapter describes how to check the validity of Oracle Communications IP Service Activator Oracle Database using the Database Integrity Checker and run clean-up datascripts.

This chapter also describes the following cleanup utilities associated with provisioning MPLS LSP based tunnels. Provisioning with the MPLS LSP module is described in *IP Service Activator VPN User's Guide*.

Checking the Integrity of a Database

You can check the validity of IP Service Activator database using the Database Integrity Checker.

> **Note:** It is a best practice to run the Database Integrity Checker nightly as part of a regular system health check.

Checking the database is particularly important if the database has been shut down in an emergency.

The Database Integrity Checker is a command-line tool that writes diagnostic information about the database to a human readable output file in *Service_Activator_Home*. By default, diagnostic information is written to the file **icheck**_*Day_Mon_dd*.**bin** but you can specify an alternative output file.

Different versions of the tool are available for different versions of IP Service Activator. You should ensure that the version of tool you use matches the version of IP Service Activator you are checking. You can do this by running the Database Integrity Checker tool and noting the version number in the first line of the output. For example:

```
Oracle Communications IP Service Activator v7.x.x.x component: Database Integrity
Checker ...
```

The tool must be run on one database at a time, and the database must not be in use.

> **Note:** Oracle recommends that you run the tool on a back-up version of the database. For information about backing up an Oracle database, see the applicable Oracle Backup and Recovery guide.

Running the Database Integrity Checker (UNIX Version)

The Database Integrity Checker (UNIX version) is installed on the Policy Server during IP Service Activator installation.

Prerequisites

- Before running the Database Integrity Checker make sure that there are no IP Service Activator components running and accessing the database.

To run the Database Integrity Checker (UNIX version):

1. Run the script **integrity_checker.sh** from the <*ServiceActivatorHome*> folder. You are prompted to enter the <*DSN_Name*>, <*database_user_id*>, and <*database_password*>.

   Enter the following commands:

   ```
   cd Service_Activator_Home
   ./bin/integrity_checker.sh
   ```

   OR

   ```
   integrity_checker -ConnectString <connect_string> [-SerialNumber <serial_
   number>] [-FileName <file_name>]
   [-Summary <file_name>]
   ```

   These parameters are described in the table "Database Integrity Checker Parameters".

   The Database Integrity Checker runs to completion and displays a report similar to the following sample:

   ```
   Oracle Communications IP Service Activator v7.x.x.x component: Database
   Integrity Checker Copyright C 20xx Oracle. All rights reserved.

   Reading database.....
   Object model initialised: 10801 objects discovered
   Checking object model.....
   Tue 07/08/07 09:31:55
   No faults were found in the database.

   Finished.
   ```

   Other reports that may be returned are listed in "Report Information".

Running the Database Integrity Checker (Windows Version)

The Database Integrity Checker runs on Windows but communicates with the Oracle Database on Solaris/Linux. The tool is installed to *ServiceActivatorHome***Program** but must be run from *ServiceActivatorHome* specifying the relative path to the program: **Program\integrity_checker.exe**

Prerequisites

- Run the Database Integrity Checker from the *ServiceActivatorHome* directory.
- Before running the Database Integrity Checker make sure that there are no IP Service Activator components running and accessing the database.

To run the Database Integrity Checker (Windows version):

1. Open a Command Prompt window.

2. Change to the *Service_Activator_Home* directory.
3. Run the tool (including the path):

```
Program\integrity_checker.exe -ConnectString <connect_string>;
[-SerialNumber <serial_number>] [-FileName <file_name>]
[-Summary <file_name>] [+debugFile -debugFileName <file_name>]
```

For example:

```
cd C:\Program files\Oracle Communications\Service Activator
Program\integrity_checker.exe
-ConnectString OCI;DSN=mydsn;UID=userid;PWD=password
-SerialNumber 123456789 -FileName DIC_Diag.out
-Summary DIC_Summ.out +debugFile -debugFileName DICdebug.log
```

The Database Integrity Checker runs to completion and displays a report similar to the following sample:

```
Oracle Communications IP Service Activator v7.x.x.xx component: Database
Integrity Checker Copyright C 2008 Oracle. All rights reserved.

Reading database.....
Object model initialised: 10801 objects discovered
Checking object model.....
Tue 07/08/07 09:31:55
No faults were found in the database.

Finished.
```

Other reports that may be returned are listed on "Report Information".

Database Integrity Checker Parameters

Table 8–1 lists and describes the integrity checker parameters.

Table 8–1 Integrity Checker Parameter Definitions

Parameter	Description
-ConnectString	Mandatory parameter. Connects the tool to the database to be checked, where *connect_string* is the database connect string. The connect string for an **Oracle database** is: `-ConnectString OCI;DSN=<local net service name from tnsnames.ora>;UID=<user identity>;PWD=<password>` In the database connection string, the value of **UID** and **PWD** should be entered in the correct case.
-SerialNumber	If you were assigned and entered a serial number during IP Service Activator installation, you must enter the same 9-digit serial number here.
-FileName	Optional parameter. Specifies the pathname of the file to which diagnostic information is sent. If the specified file already exists, a message asks if you want to create a new file or overwrite the existing file. If you do not enter the **-FileName** parameter, an output file named **icheck**_*Day_Mon_dd*.**bin** is created in the directory from which the tool was launched (for example: icheck_Mon_Apr_15.bin). **Note:** If there are no issues with your database the icheck file is not generated.

Table 8–1 (Cont.) Integrity Checker Parameter Definitions

Parameter	Description
-Summary	Optional parameter. Specifies the pathname of the file to which summary information is written. This file is appended to if it already exists. If no **-Summary** parameter is specified, summary output is displayed on the screen. Maintaining a summary file provides a useful record of when the tool was run and the results of each run.

Note: If you want to use the UNIX version of the tool (**integrity_checker.sh**), do not enter the ConnectString parameter, but you can specify the remaining parameters if needed.

Report Information

After completing the check, the Database Integrity Checker writes one of the following report messages to the screen or to a summary file, if specified:

- No faults were found in the database.

  Finished.

- A minor fault was found in the database. Please contact Oracle Global Customer Support customer support for assistance.
- A major fault was found in the database. Please contact Oracle Global Customer Support for assistance.
- The database integrity checker has encountered an unknown fault. Please update your software with the latest version of the checker.

  This message may mean that the version of the tool is incompatible with the version of IP Service Activator it has been run against.

Diagnostic Information

If the Database Integrity Checker discovers a fault, diagnostic information about the fault is either written to the **icheck**_*Day_Mon_dd*.**bin** file located on the *Service_Activator_Home* directory or the file specified by the **-FileName** parameter.

Database Administration for the Network Processor

This section describes administration and maintenance of the Network Processor portion of the Oracle database. The Network Processor use of the Oracle database complements the Policy Server's storage of the Object Model in the Oracle database.

Features of the Network Processor that use the Oracle database include:

- Device model and Service model persistency
- npAdmin tools
- Log file read processes: LogReader
- npSnapshot and npUpgrade scripts (For more information about these scripts, see the Upgrades chapter in *IP Service Activator Installation Guide*.)

Network Processor database-related files include:

- Network Processor database-related configuration files
- Network Processor database-related log files

Device Model and Service Model Persistency

The Oracle database provides the repository for IP Service Activator device configuration data. For each device managed by an IP Service Activator cartridge, there is a database record containing the device ID, the Service Model version, and the Device Model version. The record stores the last configuration successfully pushed to the device - the Last Device State (LDS).

The database record replaces the **DeviceModel_smVersion_dmVersion_componentID.xml** file for each device that was formerly maintained on disk in the directory *Service_Activator_Home*/**WorkingData/cacheDD**.

npAdmin Tools

The npAdmin shell script provides a set of tools for managing LDS records (containing deviceID, device model, and service model data) in the Network Processor database table. The default Connection String for this script is:
OCI;DSN=IPServiceActivatorDb;UID=admin;PWD=admin

The following tasks can be performed on the LDS schema in the Oracle database.

To retrieve from the database the Last Device State for a specific device:

```
Usage: npAdmin.sh get_last_lds <deviceObjid> <cartridgeObjid>
Example: npAdmin.sh get_last_lds 123 256
```

To delete one or all of the Last Device States or service models for a specific device:

```
Usage: npAdmin.sh delete_lds <deviceObjid>|all <cartridgeObjid>|all
<smVersion>|all <dmVersion>|all
Example: npAdmin.sh delete_lds 123 all all
```

The deletion of a device state is supported only in a lab environment.

To list one or all Last Device States based on device object ID or device IP address:

```
Usage: list_lds <deviceObjid>|all <cartridgeObjid>|all <deviceIp>|all
Example: npAdmin.sh list_lds 39432 all all
```

To split a device model file containing a service model into two files, a device model file and a service model file:

```
Usage: npAdmin.sh split_lds <cacheddFilename> <smFilename> <dmFilename>
<aiFilename> <directory>
Example: npAdmin.sh split_lds cachedd_123.xml sm_123_2.0.1_2.1.4.xml dm_123_2.0.1_
2.1.4.xml
```

The device model file to be split is in the current working directory, and the resulting split files are placed in the same directory.

To import a specific device model file from the specified directory into a database record:

```
Usage: npAdmin.sh import_lds <deviceObjid> <cartridgeObjid> <smVersion>
<dmVersion> <directory>
Example: npAdmin.sh import_lds 123 2.0.1 2.1.4 /opt/lds_files
```

Do not use spaces in the <directory> argument.

The files on which this command acts must have this service model filename format: **sm_<deviceObjid>_<smVersion>_<dmVersion>.xml**

and this device model filename format: **dm_<deviceObjid>_<smVersion>_<dmVersion>.xml**

To export a specific device record from the database to a file in the specified directory:

Usage: npAdmin.sh export_lds <deviceObjid> [<cartridgeObjid> <smVersion> <dmVersion>] <directory>
Example: npAdmin.sh export_lds 39432 'pwd'

Do not use spaces in the <directory> argument.

The files which this command creates have this service model filename format: **sm_<deviceObjid>_<smVersion>_<dmVersion>.xml**

and this device model filename format: **dm_<deviceObjid>_<smVersion>_<dmVersion>.xml**

To save in the specified directory the obsolete Last Device States for a specific device or all devices:

Usage: npAdmin.sh save_old_lds <deviceObjid>|all <cartridgeObjid>|all <deviceIp>|all <directory>
Example: npAdmin.sh save_old_lds all all /opt/lds_files

Do not use spaces in the <directory> argument.

The files which this command creates have this service model filename format: **sm_<deviceObjid>_<smVersion>_<dmVersion>.xml**

and this device model filename format: **dm_<deviceObjid>_<smVersion>_<dmVersion>.xml**

To delete from the database the obsolete Last Device States for a specific device or all devices:

Usage: npAdmin.sh purge_old_lds <deviceObjid>|all <cartridgeObjid>|all <deviceIp>|all [commit]
Example: npAdmin.sh purge_old_lds all all commit

Use the 'commit' option if you do not want a confirmation prompt.

To delete associations from the service and device models for a specified device:

When a service (or an instance of a service) is deleted in IP Service Activator, the Network Processor may fail to successfully remove the configuration associated with that service. This can result in repeated attempts by the network processor to remove the affected configuration.

The configured association can be removed manually or can be removed with a special **purge_association** command in npAdmin.

This command should only be used in the situation described above and at no other time. There is no 'undo' option after using this command.

To execute this command in npAdmin, the association IDs and device IP must be determined. The following log pattern in the network processor log for the cartridge provides this information:

<date-time>|<msecs>|WARN|<thread-name>|<device-ip>|<cartridge-name>|<association-list>|proxy_np|<hostname>|Error occurred while attempting to configure the device: '<command-and-router-response>'. Rollback will be attempted.
The following explains the syntax required to execute the command.

```
Usage: npAdmin.sh purge_associations [deviceIp | deviceObjid] [associations]
Example: npAdmin.sh purge_associations 10.156.68.228 25856
```

If more than one association is specified, they should be separated by a space.

Network Processor Database-related Files

The IP Service Activator administrator should be aware of the following Network Processor database-related files.

Network Processor Database-related Configuration Files

The IP Service Activator administrator should be aware of the following Network Processor/database-related files.

Service_Activator_Home/Config/cman.cfg

The entries for system_log, policy_server and event_handler include a ConnectString parameter for the OM database.

The network_processor entry includes a ConnectString parameter for the Network Processor schema.

The logreader entry includes DatabaseServer, DatabasePort, DatabaseSid, DatabaseUserid and DatabasePassword parameters for the LW schema.

Service_Activator_Home/Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor/default.properties

The following tables list attributes in the network processor **default.properties** file.

Table 8–2 Connection Settings

Network Processor default.properties attribute	Type/Values	Description
connectionTimeoutSeconds	integer	Connection timeout value once connection is established
showConfigTimeoutSeconds	integer	Timeout value once "show configuration" established
minSecondsBetweenIndividualConnectAttempts	integer	Minimum time between connection retries
overallConnectAttemptSeconds	integer	Total time to allow to attempt to connect to the device

Table 8–3 Monitoring Settings

Network Processor default.properties attribute	Type/Values	Description
warnInOfflineMaintenanceMode	boolean (true, false)	Determines whether or not to raise warning faults in Offline Maintenance (FileInterface) command delivery mode

Table 8–3 (Cont.) Monitoring Settings

Network Processor default.properties attribute	Type/Values	Description
suppressAuditFeedback	boolean (true, false)	When set to 'true', audit feedback is suppressed. When set to 'false', device and concrete audit states are updated as a result of a device audit, and faults are raised against concretes reporting audit failures.
suppressManualConfigInPerServiceAudit	boolean (true, false)	When set to 'true', manual configuration is suppressed in Per Service Audits. When set to 'false', manual configuration is displayed in Per Service Audits. Default is 'true'.
lossOfSyncSeverity	One of: Critical, Error, Warning, Notice, Info, Suppress	Configures the severity of a loss of synchronization detected between the last device state and the device

Table 8–4 Cartridge-centric Parameters

Network Processor default.properties attribute	Type/Values	Description
saveRunningConfig	boolean (true, false)	Configures whether or not the running configuration is saved to the startup configuration after each running configuration change (for applicable devices).
vrfRebuildRetries	integer	Maximum number of retries to configure a VRF routing table. This applies to cartridges that support retries to configure a VRF routing table.
vrfRebuildWaitTime	integer	Wait time in milliseconds between retries to configure a VRF routing table. This applies to cartridges which support retries to configure a VRF routing table.
manualVrfProtection	boolean (true, false)	When set to 'true', VRF routing tables are not deleted. This applies to cartridges which implement manual VRF protection. **Note**: If this value is set to 'true', then the Cisco "preCheckVrf" precheck must be disabled in the **standard.properties** file. Otherwise, it would prevent the reuse of any VRF names that had previously been preserved since the precheck would always fail.

Table 8–4 (Cont.) Cartridge-centric Parameters

Network Processor default.properties attribute	Type/Values	Description
reconcileFailureIsCritical	boolean (true, false)	Set the severity of fault message 3510: 'Re-issue commands operation failed' When set to 'true', the severity level is set to **Critical**. When set to 'false', the severity level is set to **Error**.

Table 8–5 Juniper Parameters

Network Processor default.properties attribute	Type/Values	Description
groupName	text	Specifies the group name to use when issuing commands on Juniper routers.
juniperConnectionRetryWaitIntervalSeconds	integer	The number of seconds to wait between retries to connect to a Juniper device after the connections were lost.
juniperJunosConfigurationMode	one of: configure,configure private,configure exclusive	Sets the configuration mode command for Juniper JUNOS CLI devices.

Table 8–6 Juniper JUNOS L VPN Cartridge Parameters

Network Processor default.properties attribute	Type/Values	Description
junosConfigureBroadcastMessages	boolean (true,false)	When set to “true,” broadcast message commands are configured by JUNOS SDK cartridges.
junosAutoGeneratedVrfNamePrefix	text	Specifies the prefix for auto-generated VRF Names to be used when configuring VPN using JUNOS SDK cartridges. By default it is set to “IPSA.”

Table 8–7 CISCO IOS XR Parameters

Network Processor default.properties attribute	Type/Values	Description
ciscoIosXrConfigurationMode	one of: configure, configure exclusive	Sets the configuration mode command for Cisco IOS XR devices.

Table 8–8 Cisco Transforms

Network Processor default.properties attribute	Type/Values	Description
ciscoPrefixListMinSequence	integer	The minimum ID number that can be assigned to Cisco prefix lists.

Table 8–8 (Cont.) Cisco Transforms

Network Processor default.properties attribute	Type/Values	Description
ciscoPrefixListMaxSequence	integer	The maximum ID number that can be assigned to Cisco prefix lists.
ciscoDisableDampening	boolean (true, false)	When set to 'false', the Cisco EBGP dampening configuration is generated.
ciscoSooRouteMapSequenceNum	integer	The default number used in Cisco route map order configuration.
ciscoPhbGeneratedACLType	One of: autoNamed, autoNumbered	This property is used by PHB and MQC classifications to indicated whether to create named or numbered ACL when user selects auto generate option. **Note:** Specification of a value other than 'autoNamed' or 'autoNumbered' causes the cartridge to raise a fault.
VrfAliasesDisabled	boolean (true, false)	When set to 'false', the Cisco VRF aliases configuration are generated.

Table 8–9 Database Connection Pool Settings

Network Processor default.properties attribute	Type/Values	Description
dbConnectionPoolMin	integer	Minimum number of physical connections that can be maintained by the pool
dbConnectionPoolMax	integer	Maximum number of physical connections that can be maintained by the pool
dbConnectionPoolConnectionUses	integer	Number of times a connection is retrieved/returned from/to the pool before it is closed and re-opened. This number should not be more than 1/3 of your Oracle Database "open_cursors" system parameter value or you may encounter "ORA-01000: maximum open cursors exceeded" exceptions. **Note:** See *IP Service Activator Installation Guide* for information about setting the value of open_cursors.
dbConnectionPoolConnectionTimeout	integer	Number of seconds which must pass before an idle physical connection is disconnected.
dbConnectionPoolWaitTimeout	integer	If dbConnectionPoolNoWait is false, number of seconds that must pass before a thread waiting for a connection times out.

Table 8–9 (Cont.) Database Connection Pool Settings

Network Processor default.properties attribute	Type/Values	Description
dbConnectionPoolNoWait	boolean (true, false)	If true, this attribute specifies that an error is returned if a call requires a physical connection while the maximum number of connections in the pool are busy.
dbConnectionPoolTafEnabled	boolean (true, false)	Set to true if and only if the Oracle Database TAF (Transaction Application Failover) feature is enabled
dbConnectionPoolRetryAttempts	integer	Number of retry attempts to create a connection pool after the connections were lost
dbConnectionPoolRetryIntervalSeconds	integer	Number of seconds to wait between retries to create a connection pool after the connections were lost

Table 8–10 Miscellaneous Settings

Network Processor default.properties attribute	Type/Values	Description
maxthreads	integer	Maximum number of processing threads
maxSchedulerThreads	integer	Maximum number of scheduler threads that can run concurrently.
SnapShotFileLocation	text	Filename and directory for the snapshot file, for the CM tftp server to pick up.
RestoreFileLocation	text	Directory location of the restore file, for the tftp server to pick up.
restoreCommandTimeoutSeconds	integer	Command timeout default if not specified in the restore template
useOfflineCommitOptions	boolean (true, false)	When set to 'true', changes the command delivery mode display string in the audit log from "file-interface" and "no-command-delivery" to "offline-maintenance" and "offline-test", respectively.
optimizeFirstCommit	boolean (true, false)	When set to 'true', optimize the first commit on a device after re-manage or network processor startup such that the service model to device model transformation only executes if the service model has changed.

Table 8–10 (Cont.) Miscellaneous Settings

Network Processor default.properties attribute	Type/Values	Description
queuePriority	One of: auto, low, high	Commit queue prioritization. This setting affects how the first commit for a device after re-manage or network processor startup is queued. **auto:** determine queue priority based on service model changes. This may involve database operations. **low (default):** all initial commit operations are given low priority without retrieving service model data from the database. **high:** all initial commit operations are given high priority without retrieving service model data from the database
archiveOnChangeDelay	integer	Delay in minutes, used by ConfigManagement scheduler's Archive on Change functionality.
disableEnvCheck	boolean (true, false)	If false, checks that the correct version of Saxon is found. If true, skips the check.
deleteTransientCmdReissueFiles	boolean (true, false)	Delete Command Reissue Files on startup.
tftpDeviceVisibleNICIpAddr	One of: “none”, IP address	In case the host has multiple NIC cards, specifies which IP to use. Default is “none”, meaning there is only one NIC. This affects the restore functionality, and should specify the NIC that has access to the device.
cmdExecutorDelay	integer	Specifies command delivery delay in milliseconds between commands sent to the device
cmdReissueDirectory	text	Directory used to store the preview of the command re-issue requests.
quarantineDelay	integer	Specifies the delay in seconds between quarantine invocations.
cacheCapacity	integer	Size of the memory cache, in bytes.
maximumCommitQueueSize	integer	Used to throttle the commit queue.

Service_Activator_ Home/Config/networkProcessor/upgradeTool/default.properties

```
# Full path name of the home directory  that will hold the "upgrade"
sub-directories.
upgradeHome = /opt/OracleCommunications/ServiceActivator/upgradeHome
# deviceList retrieval method = 'database' or explicit list (use one or the
other). Explicit list:
oid,name,ipa:oid,name,ipa:oid,name,ipa:oid,name,ipa:oid,name,ipadeviceList =
```

```
database
# Location of last device models to be upgraded:  'files' (if stored in cacheDD
directory), 'database' (if stored in np_lds table) or  'np_ldm' (if stored in np_
ldm table). npModelSource = database
# Checks that saxon is the correct version  as is hardcoded in the network
processor. disableEnvCheck = false
# Number of concurrent processes (should be less than the number of processors)
processes = 2
# The total amount of memory (in MB ) to be used by all children processes  (it
will be divided among them) totalMemory = 2048
```

The npSnapshot and npUpgrade read the **db.properties** file in the *Service_Activator_Home*/**Config** for database details.

Service_Activator_Home/Config/java/com/Oracle/serviceactivator/logreader/logCollector.properties

The following entries are used for configuring the LW schema:

```
dbDriver = oracle.jdbc.driver.OracleDriver
dbConnect = jdbc:oracle:thin:@SERVER:PORT:SID
maintenanceInterval = 3600000
logLifetime = 7
listenSocketPort = 4446
fileRollCheckInterval = 60000
# By default, use the enterprise edition features for efficiency
dbInitializeProperties=dbInit.properties
# For Oracle standard edition, the above line must be commented out and the
following uncommented.
#dbInitializeProperties=dbInitSe.properties
```

Service_Activator_Home/odbc/network/admin/tnsnames.ora

This file contains entries similar to the following sample, which the applications in the cman.cfg file use to connect to the appropriate database with their ConnectString parameter:

```
IPServiceActivatorDb =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = 10.13.4.113)(PORT = 1521))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = IPSA.WORLD)
)
)
```

9

Troubleshooting IP Service Activator

This chapter provides troubleshooting information for Oracle Communications IP Service Activator.

Running Troubleshooting Scripts

The following script is available to assist you in retrieving information about your deployment platform. This script is located in the IP Service Activator **bin** directory.

- ipsaps: when run, this script displays the IP Service Activator processes that are currently running.

Example: Using the IPSAPS Script to Verify which Components are Running

1. Log onto the server where you want to verify the components.
2. View which processes are running:

   ```
   $ ipsaps
   ```

3. Check, for example, that the Network Processor is started. If a process is listed that contains the word **networkprocessor**, the Network Processor is running.

   ```
   ipsaadm 13964 13763 S 31760 138176        0:14 event_handler +debugFile -debu
   ipsaadm 13778 13763 S 38640 143848        0:37 policy_server +debugFile -debu
   ipsaadm 13768 13763 S 13368 114008        0:00 system_log -ComponentName mast
      USER   PID  PPID S  RSS  VSZ        TIME COMMAND
   ipsaadm 13763     1 S 7448 8864        0:00 cman -Service no +debugFile -debu
   ipsaadm 13974 13763 S 22656 29424         0:14 integration_manager -timeout 0
   ipsaadm 13982 13763 S 56784 87120         0:55 networkprocessor
   ipsaadm 12919     1 S 5352 6440        0:02 omniNames -logdir /opt/Orchestrea
   ```

4. If a component is not running, start the component. See "Starting and Stopping IP Service Activator".
5. If problems arise, refer to install-related logs in **logs/ipsacm.log**.

Communication Errors with the Database

Communication errors may be reported when starting IP Service Activator and can indicate a problem with the connection to the database. You may see an error message displayed in the Start Component Manager dialogue box. For example, the following type of error message may appear:

```
!!!The Orchestream Policy Server has failed to start!!!
This failure is possibly due to the following:
1. OCDB or Oracle client are not set up correctly.
```

```
2. Database Username and Password are incorrect.
3. Database is not contactable from this machine.
SHUTTING DOWN
!!!!Component Manager is now shutting down and stopping components!!!!
```

This type of error can result from improper configuration of the Oracle client or ODBC Driver. Refer to *IP Service Activator Installation Guide* for configuration instructions.

Component Errors

This section provides topics to help you resolve connection issues about restarting components, reconnecting to the server, and resolving other component errors.

This section includes the following topics:

Component Will Not Restart

If a component does not start, the component manager attempts to restart it. If, after three attempts, the component fails to start, the component manager abandons the restart and the following message is displayed in the current faults pane.

```
Component bouncing: shutting it down
```

Restart the component manager to restart the failed component.

GUI Cannot Connect to Server

The error message “Could not connect to server” can occur if you attempt to log in to the user interface before you start the Policy Server. The problem generally arises if you are manually starting components on the Policy Server, and you start components and the GUI in the wrong order. If you have set up the system components to run automatically, the Policy Server automatically starts before you attempt to log in to the user interface.

> **Note:** If this error is displayed and you have installed system components to run automatically, remember that initially you must start the component manager manually. The component manager will subsequently restart automatically if the host machine is rebooted or the run level lowered and resumed.

To resolve the error:

1. Click **OK** to close the error message box.

2. Run **ipsaps** to confirm that the naming service component (omniNames) is running. If **ipsaps** is not installed, you can use the following command to confirm that omniNames is running:

   ```
   $ ps -ef | grep omniNames
   ```

3. Navigate to **/opt/OracleCommunications/IP Service Activator/bin** and run the command to start the component manager:

   ```
   $ ipsacm start
   ```

4. If the Policy Server starts successfully, start the user interface.

   If the problem persists and the same error message appears, contact Oracle GCS.

   If the Policy Server fails to start, see the discussion about manually configuring the Oracle client connection for Solaris in *IP Service Activator Installation Guide*.

GUI Hangs When Running Multiple Interfaces on Same Machine

The IP Service Activator GUI hangs when multiple network interfaces are running on the same machine. When IP Service Activator GUI hangs either turn off the extra network interfaces or add the BOAiiop_name_port option.

By default, the BOA can work out the IP address of the host machine. This address is recorded in the object references of the local objects. However, when the host has multiple network interfaces and multiple IP addresses, it may be desirable for the application to control what address the BOA should use. This can be done by using the -BOAiiop_name_port option as mentioned in the following procedure.

To resolve hanging of IP Service Activator GUI:

1. Choose which of the network interfaces you want to use to communicate with the IP Service Activator server from the IP Service Activator GUI client.
2. Note the IP address of the machine on which the IP Service Activator GUI is running.
3. Edit the "User Interface" shortcut. For this do the following:
   - From **Start** > **Programs**, go to <*ORACLE_HOME*> > **IP Service Activator** > **User Interface**.
   - Right-click on **User Interface** and select **Properties**. This brings up the User Interface dialog. On the User Interface Properties dialog select **Shortcut** tab.
4. Add the "-BOAiiop_name_port <hostname[:port number]>" option in the **Target** field; where <hostname[:port number]> is the IP address from Step 1. This options tells the BOA the hostname and optionally the port number to be used.

   For example:

   "C:\Program Files\OracleCommunications\IP Service Activator\Program\ipsa_explorer.exe" -BOAiiop_name_port 10.178.139.176

Client Cannot Connect to Server

The following error message may appear:

```
Error message: Client not authorized to connect to server
```

1. When you change users, you should export your display to the machine you are working on. Enter:

```
DISPLAY=machine_name:0.0
export DISPLAY
```

2. In a new terminal window, enter:

   ```
   xhost +machine_name
   ```

   The **xhost +** command allows other programs to use your display. For example, if you are logged in as **root**, it enables the **oracle** user to use your display also.

Component Manager Fails on Start-up

The component manager may fail while appearing to start successfully. The user interface component will therefore not start.

1. Check whether the component manager is running, using **ipsaps**.
2. If the component manager is running, go to the directory **/opt/OracleCommunications/ServiceActivator/bin** and shut it down:

   ```
   $ ./ipsacm stop
   ```

3. Delete the **cman.ini** file.

   The default path is **/opt/OracleCommunications/ServiceActivator/WorkingData**

4. Restart the component manager. For this, go to the directory **/opt/OracleCommunications/ServiceActivator/bin** and use the command:

   ```
   $ ./ipsacm start
   ```

5. If you have changed the IP address, delete it from the **omninames_*.log** files in the **/opt/OracleCommunications/ServiceActivator/WorkingData** directory.

If the problem persists, contact Oracle GCS.

Component Does Not Restart

If a component does not start, the component manager attempts to restart it. If, after three attempts, the component fails to start, the component manager abandons the restart and the following message is displayed in the current faults pane.

```
Component bouncing: shutting it down
```

Restart the component manager to restart the failed component.

Investigating a User Interface Error on Windows

If an error occurs in the IP Service Activator user interface, a Critical message is written to the current faults pane. These messages are displayed on a red background.

If you are running IP Service Activator on Windows, a message is written to the Windows Application Log (viewable with the Event Viewer).

Details of the error appear in a Problem Reports and Solutions report file. The file can be sent to Oracle GCS for investigation of the cause.

Insufficient Privilege to Stop the Component Manager or Naming Service

The component manager and naming service can only be started and stopped by the **ipsaadm** user. If you try to stop either of these processes as another user the message

Insufficient privilege is displayed. Change to the **ipsaadm** user before trying the stop command.

Transactions

This section provides topics to help the installer resolve any transaction issues that the user may be having.

This section includes the following topics:

Cannot Save or Commit a Transaction

This generally indicates connection loss between the user interface and the policy server and may be due to the fact that the policy server has failed. If the policy server is not available, the toolbar's save and commit buttons are grayed out and the relevant menu options are not available. When the connection to the policy server is restored, users can save or commit the changes they have made in the current transaction.

Reviewing Transactions

When a problem is encountered, it is helpful to review the transaction to identify the problem.

To review a transaction:

1. Click the **System** tab in the hierarchy pane.
2. Expand the **Transactions** folder.
3. Right-click on either the **Committed Transactions** or the **Scheduled Transactions** folder.

   Note: You can also right-click on the Transactions folder and perform the search at a higher level.

4. Select **Find From Here** from the pop-up menu.

   The Find dialog box appears.

5. Enter search criteria in the **Find What** field and click **Find**.

   Search results appear in the Find dialog box.

6. Right-click on the desired transactions and select **Properties** to open the transaction's dialog box.

   From the transaction dialog box, you can also review all information related to the transaction.

   Note: You can also search for transactions from the Tools menu. Select Find to open the Find dialog box.

Exporting Transactions

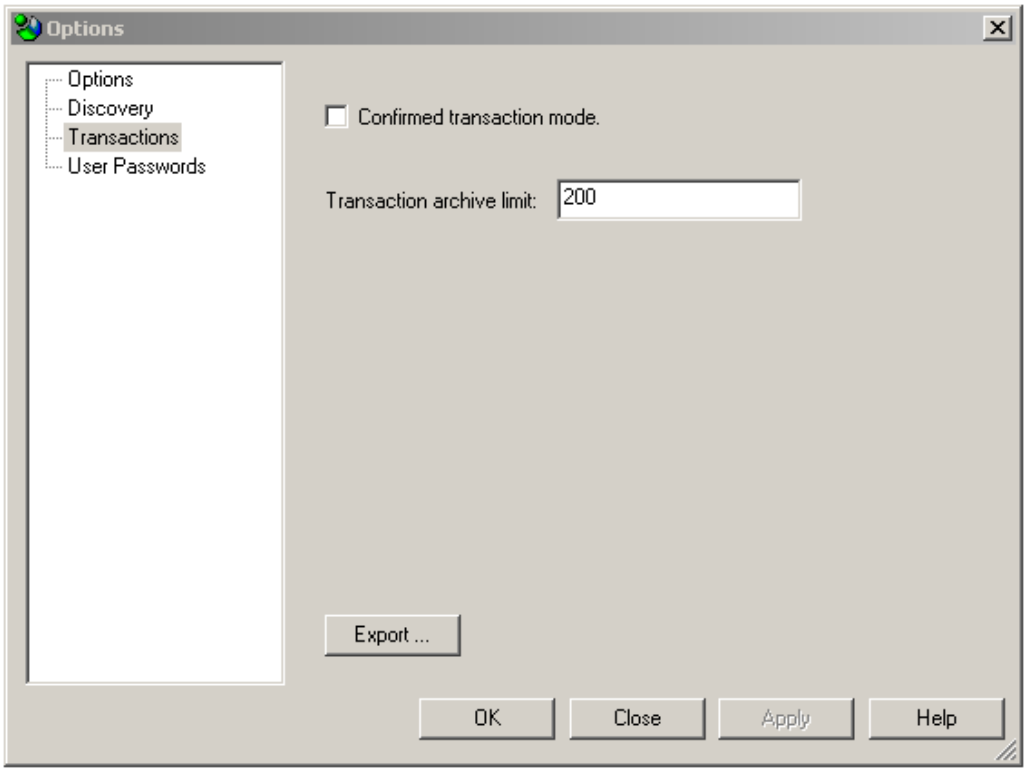
If you want to send information on the transaction to Oracle GCS, you can export the details of the transaction to a text file.

To export all transactions:

1. Select **Options** from the **Tools** menu.

   The Options dialog box appears.

2. Select the **Transactions** property page.



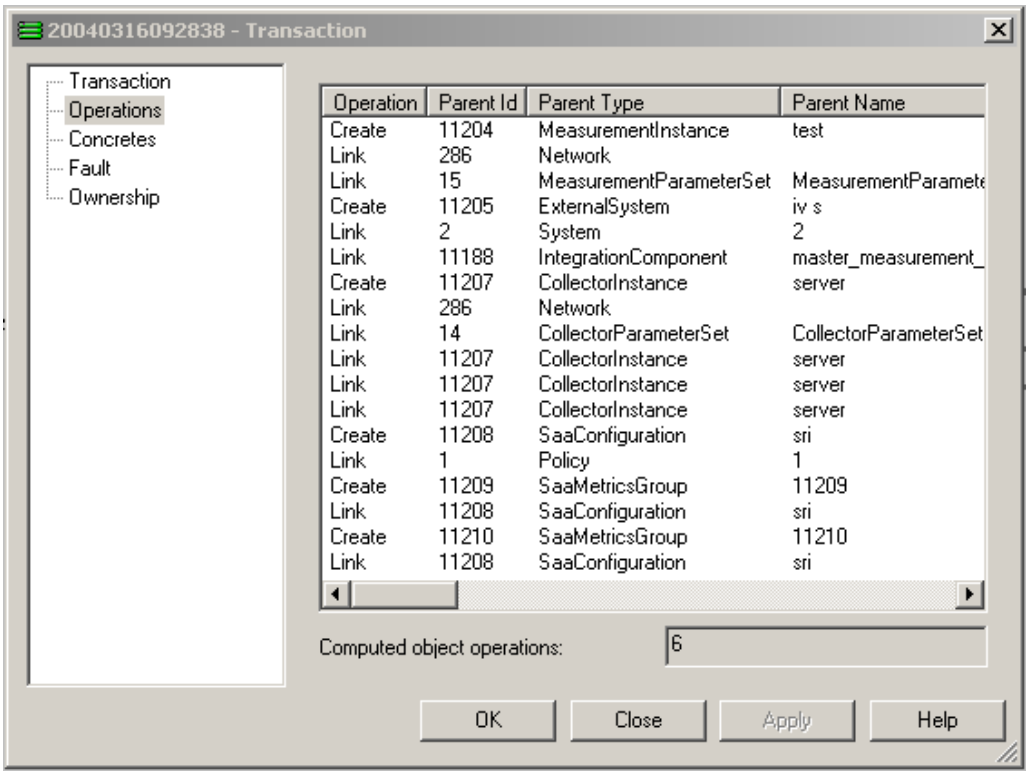
3. Click **Export** to export the transactions to a text file.

   The Save As dialog appears.

4. Enter a filename for the text file and click **Save**.

To export a single transaction:

1. Locate the single transaction that you want to export. If needed, refer to the procedure "Reviewing Transactions".
2. Right-click on the transaction in the Find dialog box and select **Properties** from the pop-up menu to open the transaction's dialog box.



3. From the **Transaction** property page, click **Export** to export the transaction to a text file.

   The Save As dialog appears.

4. Enter a filename for the text file and click **Save**.

Log Files

This section describes:

- How to generate the debug log files required by Oracle GCS
- How to collate log files on Solaris/Linux

Generating debug log files for non-Java based IP Service Activator components

This section applies to the following components:

- Component manager
- Integration manager
- Policy server
- Event handler
- IP Service Activator GUI

For details about how to set the logging level to Debug for the Network Processor, refer to "Changing the Logging Level Filter".

For more information about setting logging levels for Java-based components, refer to "Generating Debug Log Files for Java-based IP Service Activator Components".

By default, IP Service Activator outputs basic logging information to its system log file. You may be instructed by Oracle GCS to switch on more detailed 'debug' logging for one or more components if, for example:

- Configuration is not being successfully propagated to devices
- A component fails to restart after failing
- A critical error message is displayed in the user interface

Switching on debug logging for a component outputs additional information about the component's execution to a log file stored in the **DebugLogs** directory.

> **Note:** Debug logging significantly increases CPU usage on the component's host machine. You should turn on debug logging only when instructed by Oracle GCS, and turn off debug logging when the necessary information has been obtained.

Debug logging is switched on using a command-line parameter that is passed to the relevant component. The parameter can be passed to a component 'on the fly' using a component parameters utility which is supplied with IP Service Activator. This is a command-line utility that can be run against any component.

> **Note:** The component parameters utility is installed by default on both Windows, Solaris and Linux. If the utility is not installed, re-run the installation program.

When full debug output is turned on, output debugging trace log files can become very large. Parameters are also available for limiting the size of these files and for reusing the same output log file for a component when its maximum size is reached. By default there is no maximum file size/file reuse instruction set for the output log file.

You can enable debug output for any IP Service Activator component.

Log files are named according to the date on which they were started, in the format *yyyymmdd*, for example, 20040403 for the 3 April 2004. If there is more than one log file for the same day then they are successively numbered with a dash, such as **cisco.log.20040403-1**.

To enable or disable debug output for a component:

1. Do one of the following:
   - On a Windows platform:
     - Open a command window.
     - Change to the IP Service Activator **Program** directory.
   - On a Solaris/Linux platform:
     - Change to the **/opt/OracleCommunications/ServiceActivator/bin** directory.
2. Type the following:

```
ComponentParameters -ComponentName ComponentName
-set Debug.All enabled | disabled | default [-set Debug.FileMaxSize value]
[-set Debug.FileReuse enabled | disabled]
```

where *ComponentName* is the name of an IP Service Activator component as listed on IP Service Activator's **System** tab.

Table 9–1 lists the parameter definitions.

Table 9–1 Component Parameter Definitions

Parameter	Description
Debug.All enabled | disabled | default	Enables or disables full debug output for a component. Setting the parameter to **default** resets debug output to the default debug streams – that is, only critical and important events.
Debug.FileMaxSize *value*	Specifies the maximum log file size in bytes. The *value* must be 2 GB or less, but we strongly recommend limiting file size to no more than 20 MB. A value of 0 indicates there is no limit, i.e. the log file may be as large as the file system allows.
Debug.FileReuse enabled | disabled	Enables or disables file reuse. When the maximum log file size is reached the log starts again at the beginning of the file.

For example, on Solaris/Linux:

```
./ComponentParameters -ComponentName juniper-USNTW666
-set Debug.All enabled -set Debug.FileMaxSize 150000 -set Debug.FileReuse enabled
```

enables debug output for the Juniper M-series driver, restricting the size of the debug log file to 150,000 bytes (150 KB). When the maximum file size is reached, the log starts again at the beginning of the same output file.

To check the value of a specific parameter for a component:

On Windows:

1. Open a command window.
2. Change to the IP Service Activator **Program** directory and type:

   ```
   ComponentParameters -ComponentName ComponentName -get Parameter
   ```

On Solaris/Linux:

- Change to the IP Service Activator **bin** directory and enter:

  ```
  ./ComponentParameters -ComponentName ComponentName -get Parameter
  ```

The -get parameter does not work for some components such as:

- All Cartridges
- LogReader
- Discovery Server
- AL5620SAMDiscoveryServer
- CTM Server
- TransactionMonitor

To check the value of all parameters for a component:

On Windows:

1. Open a command window.

2. Change to the IP Service Activator **Program** directory and enter:

```
ComponentParameters -ComponentName ComponentName -all
```

On Solaris/Linux:

- Change to the IP Service Activator **bin** directory and enter:

```
./ComponentParameters -ComponentName ComponentName -all
```

Collating Log Files

Note: If you have identified the time at which the problem occurred, we recommend you collate only the relevant log file sections. For example, if the problem occurred at 13:20:45, send logging information for the five-minute period leading up to this time.

Solaris/Linux

Supply the log files from the following directory:
/opt/OracleCommunications/ServiceActivator/logs

Windows

Supply the log files from the following directory: **Program Files\Oracle Communications\Service Activator\DebugLogs**

For information about generating debug log files, see "Generating debug log files for non-Java based IP Service Activator components".

Enabling Debug Log Files for a Windows GUI

If you encounter any errors or issues with the Windows GUI, turn on the debug log file as follows:

1. Close IP Service Activator.
2. From the **Start** menu, go to **Oracle Communications >Service Activator > User Interface**.
3. Right-click **User Interface**, and select **Properties**.
4. In the **Target** field add the following text after the existing text:

```
"CORBA;master_server" +debugAll +debugFile -debugFileName explorer.log
```

5. Close the **Properties** window.

Refer to this debug log file if additional troubleshooting is required with Oracle GCS.

Generating Debug Log Files for Java-based IP Service Activator Components

Logging levels for IP Service Activator components using log4j are set by editing values in their properties files. These components include:

- Network Processor
- Configuration Template module
- TACC module
- Policy Services INA Integration module
- Log reader

For details about logging levels, refer to "Logging Levels". The same settings apply to all Java-based IP Service Activator components.

For details about how to set the logging level to Debug for the Network Processor, refer to "Changing the Logging Level Filter". The same principles apply for all Java-based IP Service Activator components, except you must set the level in the properties file for the particular component you are configuring.

Checking and Deleting Core Files

If an IP Service Activator component fails, a process creates a core dump file in the **logs** directory. Core dump files are named as follows:

core.*process_name*.

The following processes create a core dump file on failure:

- Integration Manager
- Policy Server
- Event Handler
- Component Manager
- Naming Service

To conserve disk space you should regularly check and delete these files.

Managing Swap File Size

When installing the IP Service Activator GUI, Windows is unable to correctly process the available swap space when the target Windows computer is configured for "System Managed" swap file size. A command window appears explaining that no swap space is available and a dialog box appears reporting an error.

To avoid this error, change the swap file management to 'Custom Setting'.

To change the swap file management:

1. Click **Start**, **Settings**, **Control Panel**, and then **System**.
2. On the **Advanced** tab, **Performance** panel, click **Settings**.
3. On the **Advanced** tab, **Virtual memory** panel, click **Change**.
4. Select **Custom size** and set a swap file size.

Discovering Manually Created Subinterfaces on Juniper M-series Devices

In order to discover manually created subinterfaces using the Network Processor on Juniper M-series devices, the **atm-option vpi maximum vcs** should be set in order to populate the ATM subinterface data into SNMP data. For example:

```
at-1/1/1 {
atm-options {
vpi 0 {
maximum-vcs 200;
 }
vpi 1 {
maximum-vcs 200;
}
}
}
```

Troubleshooting the Network Processor

This section provides topics to help the installer isolate and resolve issues during installation, setup, and operation.

> **Note:** There is no mechanism for repairing IP Service Activator components. If faults are reported by the operating system when trying to start up a system component, you should remove and reinstall the software.

This section includes the following topics:

Viewing Debug Logs

One of the first steps to solving a problem is to view the logging at a more detailed level. The default logging level is “Info”, which provides a medium logging level. You can open up Network Processor logging for debug purposes by performing the procedure "Changing the Logging Level Filter".

Then you can use the following procedure to display debug logging for the Network Processor component while performing other Network Processor-related activities.

1. To display the log output on the screen, enter:

   ```
   $ tail -f /opt/OracleCommunications/ServiceActivator/logs/networkprocessor.log
   $ tail -f
   /opt/OracleCommunications/ServiceActivator/AuditTrails/np<cartridgeName>.audit.
   log
   ```

   Replace <*cartridgeName*> with the name of the cartridge.

2. To exit the log file, press **Ctrl+C**.

Network Processor Cannot Find a Cartridge

If the cartridge does not appear in the **Drivers** folder under the Network Processor in the **Hierarchy** pane (and on the **Drivers** property page of the **Network Processor** dialog box), or if the cartridge appears in the **failed** state, perform the following steps:

1. You may not have installed the cartridge jar file. Check that the cartridge jar file is present in the **/opt/OracleCommunications/ServiceActivator/lib/java-lib/cartridges/** directory.
2. You may not have installed the latest version. Check the version/date of the cartridge jar file to ensure that the current version has been installed.

3. The cartridge file might not have been detected by the Network Processor. Stop and restart the Network Processor.
4. Verify that the Network Processor and cartridge are now functional by checking the GUI screens:
   - Right-click the <*NetworkName*> on the **Domains** tab and select **Properties**. The **Network Processor** dialog box shows the expected Network Processor state and restart time.
   - Check the **Drivers** property page. The cartridge should be listed as **cartridgeName-<hostname>**, for example, Huawei-srvotlab481.
   - In the **Drivers** folder under the Network Processor object, right-click the cartridge object and select **Properties**. On the **Cartridge** dialog box (the **cartridgeName-<hostname>** dialog box), ensure that the cartridge is in a running state.

Turning Off the Offline Maintenance Mode Warning

When a device object is put into Offline Maintenance mode in the GUI, or the Network Processor is running with -FileInterface enabled, any IP Service Activator-generated configuration commands intended for that device may not actually be sent to the device.

A warning is issued for each affected concrete:

```
Message 3311: Changes to configuration were attempted while the device was in
Offline Maintenance mode; some configuration has not been applied to the device.
```

The parameter **warnInOfflineMaintenanceMode** can be set using the Configuration GUI to control whether these warnings are issued.

See "About the warnInOfflineMaintenanceMode Parameter".

Command line and ComponentParameters for Network Processor

The command line and ComponentParameters for Network Processors must recognize the FileInterface and NoCommandDelivery options. They must also recognize OfflineMaintenance and OfflineTest options. The same set of rules apply to the new preferred options. The values are boolean (true, false), and are enabled when set to true, and disabled when set to false.

For more information about command delivery modes, see "Monitoring and Managing the Network Processor".

Setting Severity of Fault “Re-issue commands operation failed”

The fault message *3510 DeviceID: Re-issue commands operation failed* is generated if commands were not successfully delivered to a device, following a Configuration Audit - Reissue commands operation.

You can modify the severity level of this fault message. To do this, edit the parameter **reconcileFailureIsCritical** in the network processor **default.properties** file, as follows:

- **True** (default value) sets message 3510 severity level as **Critical**. When message 3510 is raised with severity **Critical**, the device is put into **Intervention Required** state. In this state, the device no longer accepts configuration commands.

- **False** sets message 3510 severity level as **Error**. When message 3510 is raised with severity **Error**, the device is put into **Down** state. In this state, the device can still accept configuration commands.
- If the parameter **reconcileFailureIsCritical** is omitted from the default.properties file, the default setting **True** (Critical) is applied to message 3510.

The default.properties file is in *Service_Activator_Home*/**Config/networkProcessor/com/Oracle/serviceactivator/networkprocessor.**

(For more information about the Configuration Audit in the IP Service Activator GUI, refer to Online Help topic **Device properties - Audit/Migrate page**.)

Recovering from Rollback Failure

This procedure describes how to recover from rollback failure on a cartridge-managed device. Rollback failure can occur when manual configuration exists on a device. When rollback failure occurs, the device is put into Intervention Required state. No additional configuration can be sent to the device until you clear the manual configuration that is causing the problem. You should restore the proper configuration on the device so that its state can change back to Installed.

Pre-requisites: Only experienced IP Service Activator users with in-depth knowledge of device configuration commands should perform this procedure, or use the Command Re-issue feature.

A feature is provided to show the commands that were not sent due to failure on rollback. Such command lines are marked in the audit log file with the prefix [unsent-command].

To recover from rollback failure:

1. Look in the Network Processor audit trails for this device. Find the entries with the “unsent-command” prefix.

   The last sent command entry before the block of unsent-commands is the one that caused the failure.

2. Note the full command string that failed, and its context (for example, the sub-interface).

3. Note the device response in the fault text. If required, get the complete response from the Network Processor log. (Again, search for “unsent-command”; the rollback failure will precede the block of unsent-commands.)

4. Based on the command being sent, and on the nature of the failure, determine the conflicting manual configuration that exists on the device, that caused the failure.

   - For example: the rollback cannot remove a service policy applied by IP Service Activator because a user has manually re-used the same policy on another interface.
   - It may be helpful to perform a device audit.
   - A better approach is to get the running configuration from the device. This step requires knowledge of the language and form of the vendor’s device configuration.

5. After you identify the conflicting manual configuration, either remove the conflicting configuration or remove the conflict.

   So, in the example given in step 4:

   - either remove the reference to the policy on the non-managed interface,

- or clone the policy manually and change the reference on the non-managed interface to the cloned policy.

6. When the conflicting manual configuration has been resolved, you can decide which of the following methods to use to synchronize the IP Service Activator device model with what is actually on the device.

   Note: Command Re-issue allows you to send missing device configuration commands independently. If additional device configuration is required, you must manually re-issue the commands. Do not attempt to use Command Re-issue in this case.

   - Run a device audit and see what commands are identified as missing. If the missing commands can be re-sent in isolation (without any pre-requisite commands other than context establishing commands), use the **Command Re-issue** feature on the **Device properties - Audit/Migrate page** to resend the missing commands. For more details, follow the procedure "Re-sending commands marked as MISSING in the Configuration Audit" in the Online Help.
   - Look at the unsent-command list to determine which of the commands must be issued in order to allow the missing commands to be re-sent, and, to identify commands that are either no longer in the audit or are present but flagged as Conflict and need to be removed. All commands from the unsent-command list must be manually re-issued to the device.

     Note: It might be simpler to manually re-issue all unsent-commands in the list and then run a device audit to see what commands are reported as missing. If there are any missing commands, then these can be re-sent using the Command Re-issue feature (if appropriate) or manually. However, this approach may churn the device configuration more than necessary.

7. Repeat this procedure until a clean audit response is returned.
8. Delete the Critical fault against the device.

   The device is restored to its previous Online state and provisioning can resume.

Network Processor Does Not Start

The Network Processor may fail to start for reasons such as errors in the cartridge registry, options, or synonym files, missing cartridge files, missing error message files.

Some of the primary reasons are:

CORBA handshake delay time

The component manager waits a configurable amount of time for a CORBA handshake from the network processor to indicate its readiness. If the handshake is not completed in that time, the component manager restarts the network processor.

Cartridge handshake delay time

A large number of complex cartridges may require the network processor startup time to exceed the default wait time (300 seconds) before the component manager checks

for the handshake. Increasing the component manager startup wait time before checking for the handshake allows the network processor additional time to complete its startup.

To change the component manager startup wait time, edit the **cman.cfg** file in the directory **/opt/OracleCommunications/ServiceActivator/**. Change the default value to a higher value.

An example from the **cman.cfg** file follows:

```
#NetworkProcessorEntry
/opt/OracleCommunications/ServiceActivator/bin/networkprocessor "-ComponentName
proxy-np-srvotlab452 -ComponentLocation srvotlab452" 1 240 1 0
```

Credentials expiry

The Network Processor does not start when the database credentials have expired or are about to expire. In this case, the Network Processor displays one of the following SQL exceptions:

- ORA-28002: The password will expire within %s days
- ORA-28001: ORA-28001: The password has expired

The Oracle database alerts the database user to change the password after its expires, and enters a grace period, which is a default value of 10 days. During the grace period the password must be changed.

There are two ways to prevent this:

- Set the PASSWORD_GRACE_TIME to UNLIMITED. A warning is issued but the connection to the database can still be made.
- Change PASSWORD_VERIFY_FUNCTION to NULL for the specific user profile.

However, these two methods are **not recommended**. They are not part of a strong security policy as they do not encourage the user to change the password. It is recommended that the database user change the password within the defined expiry time for the specific database user profile.

VMSize

The Network Processor may fail to start when the vmsize (defined in startup script) is set too small with the xmx value set to lower than 3.5 GB.

If Concrete Rules Are Not Created for Abstract Rules

If no concrete rules are listed for an abstract rule, check that:

- the correct device and interface roles have been associated with the rule and assigned to the relevant policy targets.
- devices to which the rule should apply are managed by IP Service Activator.

Running Troubleshooting Scripts

The following script is available to assist you in retrieving information about your deployment platform. This script is located in IP Service Activator's **bin** directory.

- **ipsaps**: When run, this script displays the IP Service Activator processes that are currently running.

Example: Using the IPSAPS script to verify which components are running

1. On the server where you want to verify the components, log in as the IP Service Activator administrator (**ipsaadm**).
2. View which processes are running:

   ```
   $ ipsaps
   ```

3. Check, for example, that the Network Processor is started. If a process is listed that contains the word **networkprocessor**, then the Network Processor is running.

   ```
   ipsaadm 13964 13763 S 31760 138176        0:14 event_handler +debugFile -debu
   ipsaadm 13778 13763 S 38640 143848        0:37 policy_server +debugFile -debu
   ipsaadm 13768 13763 S 13368 114008        0:00 system_log -ComponentName mast
      USER   PID  PPID S  RSS  VSZ        TIME COMMAND
   ipsaadm 13763     1 S 7448 8864        0:00 cman -Service no +debugFile -debu
   ipsaadm 13974 13763 S 22656 29424         0:14 integration_manager -timeout 0
   ipsaadm 13982 13763 S 56784 87120         0:55 networkprocessor
   ipsaadm 12919     1 S 5352 6440        0:02 omniNames -logdir /opt/Orchestrea
   ```

4. If a component is not running, start the component. See "Starting and Stopping IP Service Activator".

If problems arise, refer to install-related logs in **logs/ipsacm.log**.

WebLogic Patch Update Errors

If you are using WebLogic Server for your IP Service Activator installation, you might receive errors after you upgrade to the latest patch version. These errors occur because the updated .jar files no longer have the correct security permissions.

> **Note:** The errors and procedure included in this section are for WebLogic Server 11*g*. Oracle recommends that you review the documentation for the WebLogic Server version that you are using before you implement this procedure.

The following is an example of the WebLogic Server security error that you might receive after applying a patch update:

```
Exception in thread "main" oracle.security.jps.JpsRuntimeException:
java.lang.ClassNotFoundException:
oracle.security.jps.internal.trust.TrustServiceProvider
        at
oracle.security.jps.internal.core.runtime.ContextFactoryImpl.findServiceProvid
er(ContextFactoryImpl.java:126)
        at
oracle.security.jps.internal.core.runtime.ContextFactoryImpl.findServiceInstan
ce(ContextFactoryImpl.java:137)
        at
oracle.security.jps.internal.core.runtime.ContextFactoryImpl.getContext(Contex
tFactoryImpl.java:170)
        at
oracle.security.jps.internal.core.runtime.ContextFactoryImpl.getContext(Contex
tFactoryImpl.java:191)
        at
oracle.security.jps.internal.core.runtime.JpsContextFactoryImpl.getContext(Jps
ContextFactoryImpl.java:129)
        at
```

```
oracle.security.jps.internal.core.runtime.JpsContextFactoryImpl.getContext(Jps
ContextFactoryImpl.java:124)
        at
oracle.security.jps.internal.core.runtime.JpsServiceLocatorImpl.lookup(JpsServ
iceLocatorImpl.java:55)
```

To give the .jar files the correct security permissions:

1. Open the *WLS_home*/**server/lib/weblogic.policy** file in a text editor.

   where *WLS_home* is the base directory for the WebLogic Server core files.

2. Add the following text to the end of the file:

   ```
   grant codeBase  "file:MW_home/patch_wls1033/patch_jars/*"
   {
         permission java.security.AllPermission;
   };
   ```

   where *MW_home* is the location where the Oracle Middleware product is installed.

Effects of Component Failures

This section provides an overview of the events that occur if an IP Service Activator software component fails, or the machine hosting one or more components fails. The effects of these failures on other system components are described.

Component failure can be divided into two areas:

- **Software failure**: An IP Service Activator component fails
- **Hardware failure**: The machine hosting one or more components fails

For information on backing up and restoring IP Service Activator data, see "Backing Up and Restoring Data".

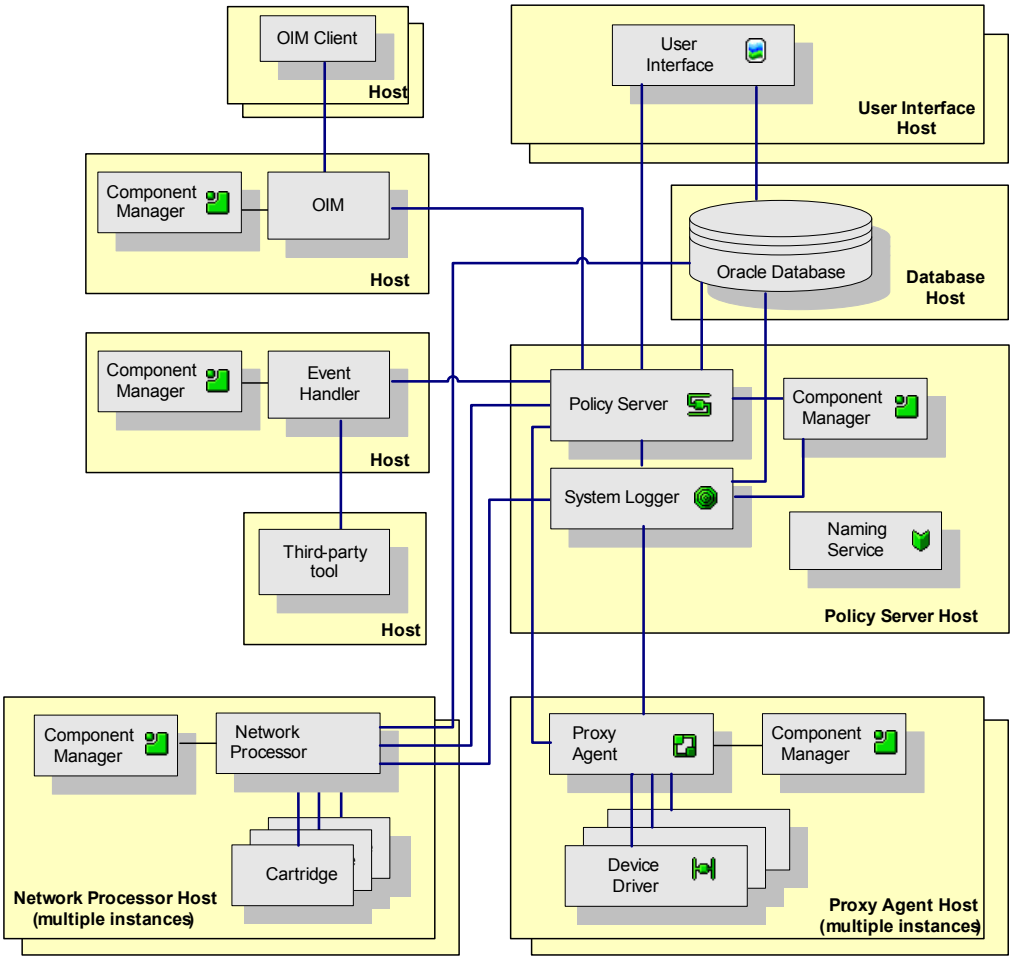
Component Distribution and Communication

In a standard deployment, components are distributed across multiple host machines. Components communicate with each other using CORBA (Common Object Request Broker Architecture) with the CORBA naming service coordinating inter-component communication.

An Oracle Database stores the system, topology and policy data generated within the system. The policy server and user interface components communicate with the database.

Figure 9–1 outlines the distribution of major components and shows the connections between them. All connections except those to the Oracle Database use CORBA.

Figure 9–1 Standard Configuration of Component Distribution and Communication



Note: All IP Service Activator components communicate with the naming service and these connections are therefore not shown.

Software Failure

This section describes the events that occur if an IP Service Activator component fails.

Naming Service

The naming service maintains CORBA name and location details for all other system components. If run as a service, it starts up automatically on installation or system restart.

As each components starts up, it registers with the naming service, passing its name and the port on which it will listen for communication from other components. When one component needs to contact another, it contacts the naming service for that component's contact details.

On Windows, an associated program called the naming service wrapper allows the naming service to be started as a system service and allows the correct environment to

be set. On Solaris/Linux, these functions are performed when you run the **ipsans** script.

> **Note:** For a list of IP Service Activator process names on Solaris/Linux see "IP Service Activator Process Names on Solaris/Linux".

Once a component has retrieved contact details for a particular component from the naming service, it maintains that information and does not need to contact the naming service for subsequent communication. If a component fails and restarts, however, it registers a new listening port. Calling components retrieve the new contact details from the naming service the next time they try to communicate with the component.

The naming service is therefore only accessed when:

- Components register with the naming service at startup
- A registered component fails and restarts and re-registers with the naming service
- One component contacts another component for the first time or after a component has failed and restarted

The naming service maintains a record of registered component details in the **omninames-***machine_name***.log** file in the **WorkingData** directory. On restarting after a failure, the naming service reads in the component details held in the file.

Effects of Failure

If the naming service is run as a service, it may fail and restart automatically before any components have tried to make contact. In this case, the failure is imperceptible.

A component may try to retrieve another component's contact details while the naming service is down. To the calling component it will appear that the target component is down as its contact details are unavailable. The calling component makes periodic calls to the naming service, however, and, if the service returns, contact details will be obtained with minimal effect on the system.

Component Manager

The component manager starts all system components, monitoring and reporting their status and restarting any that fail. It raises faults on components and updates the system logger with component status information – that is, whether the component is up or down.

On restarting after a failure, the component manager checks the lock files in the **WorkingData** directory. This file records which processes were running when the component manager failed. The component manager tries to connect to these processes. If it is unable to make a connection, the component manager restarts the relevant components.

Effects of Failure

If the component manager fails:

- Any components that fail are not restarted automatically
- Users cannot execute the **Shutdown All** or **Startup** command from the user interface to shut down or start up components on the host machine
- Information on component states is not reported to the system logger

System Logger

The system logger receives messages from the policy server and proxy agent and details of component status from the component manager. It stores the information that it receives in the system database.

Effects of Failure

If the system logger fails, proxy agent and policy server faults and component status details are not passed on to the database and they are silently dropped.

Policy Server

The policy server is the central component of IP Service Activator. It coordinates access to the database from the user interface components, performs network topology discovery, and validates and transmits policy configurations to the proxy agents.

The policy server maintains the current state of the system, stored in the database – operations with the database are transaction controlled to maintain consistency.

Effects of Failure

If the policy server fails while saving a transaction to the database, when it restarts it returns to the system state as it was before the most recent successful save.

By default, data that is saved by a user is queued by the policy server before being saved to the database. If there is a large number of users accessing the database, users may lose changes that appeared to be saved in the user interface. Running the user interface in **confirmed transaction mode** reduces the likelihood of lost data, but slows a user's progress in the user interface. In this mode, changes cannot be made in the user interface until the policy server notifies the component that a transaction's changes have been saved to the database. For more information on confirmed transaction mode, see *IP Service Activator User's Guide*.

If the policy server fails while propagation is in progress, on restart it re-propagates the most recent saved configuration to the proxy agents. This may take some time to complete.

If components try to contact the policy server while it is down, they continue to attempt contact until the policy server is back on line. An error message is logged by the components reporting on the loss of connectivity.

Users may enter changes via the user interface while the policy server is down but may not save or commit transactions. When connectivity is reinstated, the object model maintained by the user interface component is synchronized with that maintained by the policy server and changes may be saved to the database.

Network Processor

The network processor manages communication to and from devices through cartridges. Within the context of IP Service Activator, it performs the conversion of user changes to the configuration of IP Service Activator objects into a set of CLI commands for delivery to devices.

The network processor and cartridge architecture is extensible and scalable. Support for new services and devices can be added by creating and deploying new cartridges and cartridge components.

Within the context of Configuration Management, the network processor manages the flow of information from a configuration policy (or configlet) to a device, again by

performing the conversion of configuration commands into a set of CLI commands for delivery to the device.

Effects of Failure

When the Network Processor fails, the component manager raise a fault and attempts a restart.

In-process audits, backups, restores, and repairs are not carried forward and will not resolve - you will have to re-submit them.

Service changes which were not yet in the communication phase, will be eventually pushed when the Network Processor comes back online.

If the failure occurred during a period when a device was actually being configured, the commands will be re-issued. For devices which ignore configuration commands that are re-applied, you might see faults indicating a write-failure to the device if the device reports back a warning that the configuration already existed. These can be ignored since the original commands were applied, and the re-application does not change any configuration settings on the device. On transaction-oriented devices (such as some Juniper devices) this behavior will not occur.

If there is not enough virtual memory allocated to the Network Processor, you could experience a series of failures and restarts. This can also be caused by an incorrect Network Processor configuration (e.g. missing cartridge files, message files, etc.)

If you see repeated failures, check the logs.

If you are having trouble getting the affected Network Processor to restart, consider moving your devices to another Network Processor.

Log Reader

The log reader collects logging information from multiple IP Service Activator components and stores it in a normalized format in the Oracle Database. The log schema is compliant with log4j.

Any third-party product which supports log4j can be used to interact with the logging data.

Effects of Failure

The log reader can scavenge logging information that was created while it was unavailable. As long as the individual component log files have not rolled over, there should be no loss of logging information.

User Interface

There is no component manager installed on the user interface host machine. Therefore, the user must restart the user interface if it fails. On restart, the user interface retrieves information about the object model from the policy server.

Effects of Failure

Any transactions that were not saved before the component failed are lost.

Event Handler

The event handler collects, filters and delivers details of faults and other events occurring anywhere in the network managed by IP Service Activator and can notify external systems of these events. The events could be, for example, a new device has

been discovered, a site has been put into a VPN, or the ability to provision a device has been lost.

The event handler can be configured to output information to a file, through SNMP traps, or through the CORBA API.

The event handler processes each active event collector and checks the events that have occurred within the collector's scope against those that have already been forwarded to the relevant third-party systems. Any outstanding events are sent to the third-party systems.

Effects of Failure

If the component fails, on restart it retrieves the EOM and details of subscribed events. This information is stored in the database or the **SubscribedEvents.log** file in the **WorkingData** directory, depending on whether a direct connection has been set up between the event handler and the database.

Events which occur while the event handler is down will not have their associated notifications sent to the third-party, external systems, nor will these messages be sent once the event handler is available again. This could have different implications depending on the external system and the type of event.

OSS Integration Manager (OIM)

The OIM provides an API to Operational Support Systems (OSSs), enabling IP Service Activator to be integrated with third-party software, such as billing, monitoring and fault management systems. The OIM enables automated or programmatic control of IP Service Activator's service provisioning facility.

If the OIM fails, the component manager automatically attempts to restart the component:

- If the policy server is running (and it is the correct version), the OIM starts and connects to the policy server.
- If the policy server is not running, the OIM attempts to connect to the policy server every five seconds.
- If an incorrect version of the policy server is running, the OIM fails to start and does not make any further attempt to connect to the policy server. However, the component manager continues to restart the OIM and only stops its automatic restart of the component if the OIM fails three times.

Effects of Failure

If the OIM fails, the Command Line Interface (CLI) gives no immediate indication of the failure. However, when the next command is entered, the CLI recognizes that the OIM session that it was connected to is no longer available. The CLI attempts to reconnect three times. On reconnection, the OIM prompts for the user name and password. If the CLI fails to reconnect after three attempts, the CLI closes.

Any scripts that are running when the OIM fails will stop.

CTM Server Component

The CTM Server acts as a back-end repository for Configuration Templates for CTM clients.

When the CTM Server is unavailable, CTM clients cannot perform CTM work.

Any instantiations of templates that are already in the IP Service Activator system are still processed and their commands pushed onto the target devices.

Note: There is a provision in the Configuration GUI to manually set the CTM server port.

Tacc Module Server Component

The TACC module server component acts as a centralized collector for all TACC requests and responses. Whenever a network-touching component crosses a threshold, it sends a confirmation request to the TACC module server.

The TACC server will then relate information to all TACC clients that have subscribed to the server. Approval or denial of the request are passed from the client back to the server and then back to the originator to accept or deny the action.

Effect of Failure

No TACC services (such as subscriptions) are available while the server is not running. The TACC clients will raise errors.

Network processors cannot contact the TACC server and revert to thresholding functionality instead of using TACC functionality, and immediately fail the transaction in progress at the time of the TACC server failure.

Hardware Failure

In most cases, there are multiple IP Service Activator components on a host machine. The exception is the user interface host, where only the user interface component is installed. A hardware failure therefore generally causes a number of components to fail simultaneously. This raises the issue of component startup order.

About the Component Manager and the Restart of Managed Components

On all hosts, if set to start automatically upon system start, the component manager restarts when the host is restarted. The component manager will also restart any managed components on the host on which it resides.

If the component manager is not set to start automatically upon system start, it will have to be manually restarted. It will then restart its managed components.

Policy Server Host

The naming service and component manager restart automatically if run as services. Otherwise they must be started manually. The component manager subsequently restarts the policy server and system logger.

User Interface Host

The user interface component is the only IP Service Activator component running on the host machine. Therefore, if the host fails there are no component startup order issues to be considered. The user interface component's behavior is described in "User Interface".

OIM Host

The component manager, when restarted, restarts the OIM.

Event Handler Host

The component manager, when restarted, restarts the event handler.

10

Improving IP Service Activator Performance

This chapter explains performance considerations and tuning techniques that can improve the performance of Oracle Communications IP Service Activator in a particular deployment.

Performance Considerations

This section discusses important performance considerations to keep in mind while planning how to deploy IP Service Activator.

Note: You must regularly monitor system resource usage in order to ensure ongoing system performance and stability.

Table 10–1 highlights various network and hardware parameters and how they affect various IP Service Activator components.

Table 10–1 IP Service Activator Network and Hardware Parameters

Metric	Most Influential Component	Influential Attributes	Other Factors
GUI commit times	GUI workstation	CPU speed	Available RAM, workstation process load
End-to-End response times - single device	Policy Server	CPU speed	Proxy Server CPU speed, Proxy Server RAM
GUI login time	GUI workstation	CPU speed, sufficient RAM	Policy Server CPU speed
OIM commit time	OIM Server	CPU speed, sufficient RAM	Policy Server CPU speed
End-to-End response times - global change (configuration to all devices)	Proxy Server or Network Processor	CPU speed, sufficient RAM	Policy Server CPU speed, Oracle Database Server resources, network speed
Time to restart Proxy Server	Proxy Server	CPU speed, sufficient RAM	Policy Server CPU speed
Time to restart Network Processor	Network Processor	CPU speed, sufficient RAM	Policy Server CPU speed, Oracle Database Server resources, network speed
Time to restart Policy Server	Policy Server	CPU speed, sufficient RAM	Oracle Database Server resources, network speed

Running Troubleshooting Scripts

When the memory requirements of the IP Service Activator components are larger than real memory, responsiveness and throughput drop significantly. The IP Service Activator processes using virtual memory behave as reliably as if residing in real memory, but suffer a significant degradation in performance. The most notable examples are the time required to start the Proxy Server, and the time required to propagate a system-wide (global) configuration change on a Proxy Server or Network Processor.

In the case of the global configuration change, the time required to propagate in a swapping system is up to eight times longer than that of a system residing in real memory. The capacity and efficiency of the network through which the IP Service Activator components are connected will also have an impact on the performance of the IP Service Activator deployment.

When determining how many Proxy Servers and/or Network Processors will be required to effectively support a network, attention must be paid to the memory footprint of the IP Service Activator processes on each server. See **Hardware requirements** in *IP Service Activator Installation Guide* for network ranges, and contact Oracle Global Customer Support for assistance.

IP Service Activator can use large amounts of system memory. It is important to be aware of operating system limits regarding the memory footprint of a single process. Under Solaris (versions 8, 9, and 10 - 32 bit applications), the per-process memory limit is 4 GB. This may impact the design of a very large system with respect to the number of Proxy Agents or Network Processors deployed.

Even though the Network Processor instances can be configured to access a database schema different than that of the Policy Server, there is no performance benefit in deploying in this fashion. The Network Processor instances of an IP Service Activator deployment should share the database schema (userid and password) with the Policy Server of that deployment. Failure to do so can result in the two schemas becoming out of synchronization, and may impede newer IP Service Activator features which logically join data from the Network Processor components of the data model with those of the Policy Server.

Tuning Techniques

Careful bench-marking should be conducted to evaluate the relative merit of each tuning adjustment.

Deploying on Solaris

Impacts: Improved IP Service Activator throughput.

Reasoning: The threading libraries in Solaris have improved, which has a positive impact on the performance of both the Policy Server and the Network Processor.

Re-ordering IP Service Activator Component Initiation

Impacts: Reduced system start-up times on systems where the Policy Sever and Network Processor are co-located on the same host.

Reasoning: At IP Service Activator system start-up, the Policy Server will load the Object Model and then 'push' those portions of the model that pertain to each individual Network Processor. This Proxy Push stage is what consumes most of the

start-up time, and after a Proxy Push begins, another cannot initiate until the current one has completed.

With this in mind, it is advantageous to ensure that all Network Processor instances in an IP Service Activator deployment have started up, and are ready to receive a Proxy Push, prior to the Policy Server beginning its Proxy Push stage of start-up. This ensures that all Network Processors receive the Proxy Push data simultaneously.

Action: Move the Network Processor start-up command line, in the Component Manager configuration file, such that it is listed above the Policy Server start-up command line (in the same file). The component Manager configuration file can be found at **/opt/OracleCommunications/ServiceActivator/Config/cman.cfg**. The Network Processor part of this file begins with the phrase *#NetworkProcessorEntry* and ends with the phrase *#End*. Simply cut the phrases, and all the content between them, and move it to just above the phrase *#PolicyServerEntry*. No cman.cfg file editing is required on machines or containers that are only hosting the Network Processor component.

Reducing the Number of Stored Transactions

Impacts: GUI Login, Policy Server Startup, and OM Memory Consumption

Reasoning: Every committed transaction performed by IP Service Activator (either using the GUI or the OIM), is stored in the Object Model, and hence the database. Over a short period of time, this store can grow very large. By reducing the number of saved transactions, the Object Model can be loaded faster as both the GUI and the Policy Server require less memory with which to store their copies of the Object Model.

Action: Modify the Transaction Archive Limit.

From the **Tools** menu, select **Options** to access the Options dialog box. In the **Transaction archive limit** field, specify a new archive limit. Set the value to 200 (default) or less.

See "Setting the Archive Limit for Transactions" for more details.

Reducing the Impact of Discovery Actions on the IP Service Activator GUI

Impacts: The responsiveness of all active clients (GUIs) during a network discovery operation.

Reasoning: When a new object is discovered, the Policy Server sends an Object Model update notification to all active clients (both GUI and OIM). If an active GUI has a Map pane displayed, and the new object(s) resides in that map, the GUI will be unresponsive for up to 60 seconds while the Map is updated.

By working within the detailed or compact device lists for a domain, customer, or VPN objects the Object Model can be updated and the list regenerated, but in a much shorter time (2 to 4 seconds).

The impact on GUI responsiveness not only affects the GUI that launched the discovery operation, but all active GUIs.

Action: Avoid displaying map panes unless there is an operational need to do so. Detailed or compact lists can accommodate most workflow requirements.

Apply Policies at the Highest Granularity Possible

Impacts: GUI performance, transaction propagation time.

Reasoning: All the clients (GUIs and OIM clients) are related to each other through the Policy Server. When one client updates and commits an object model change, the object models on each of the other clients must be updated also. When the object model update occurs, it can manifest itself as the GUI not responding, or appearing to be 'frozen'. Observing the CPU usage of the workstation (using the Task Manager,) will confirm that the GUI is functioning, but that it is consuming CPU cycles as it updates its internal Object Model.

More granular policies reduce the number of devices impacted by a configuration change, and thus reduce the impact on the Policy Server and cartridges.

Comparing the extremes (policy applied to a device versus a domain), the difference in propagation time can be up to two orders of magnitude.

Additionally, policy applied at the device level causes update and commit commands to be set only to the Network Processor that manages the target devices. While policy applied at the domain, customer, or network level causes the same commit commands to be propagated to all Network Processors in the IP Service Activator instance.

The transaction is not complete until all Network Processors have processed the commit commands. Applying policy at higher levels of the hierarchy, than the device, impacts the throughput of the IP Service Activator instance.

Action: In order to mitigate this behavior it is recommended that policies be applied at the device level as much as possible. This will reduce the scope of any object model change requests in the clients and at the cartridges.

Reducing the Component Manager Shutdown Timeout

Impacts: The length of time that the Component Manager will wait, after the shutdown signal has been sent, on devices to shutdown gracefully before killing the processes.

Reasoning: The default component manager shutdown timeout period is 60 seconds (1 minute). That means that the component manager will wait for 16 minutes after first initiating a component shutdown before forcefully shutting down any components that have not done so gracefully. Most components will take far less time to shut down if they are able to do so and it should not take 16 minutes to determine that a component must be shut down forcefully. By reducing this time period, a system can be shut down faster.

Conversely, forcefully shutting down components that are attempting to shut down gracefully will result in program core files being generated.

Action: Leave the default setting unless there is a strict need for a reduced system shutdown time. If faster shutdown is required, reduce the ShutdownTimeout setting using the Configuration GUI tool from 960 seconds to a range between 210 and 480 seconds.

See "Setting the Component Manager Shutdown Timeout Period" for more information.

Increasing the ORB TCP Timeout Threshold

Impacts: Increases the amount of time that the Configuration Management Server can wait for a response from the OSS Integration Manager (OIM) when conducting a device search on a very large IP Service Activator network.

Reasoning: When a network managed by IP Service Activator becomes very large (over 10,000 devices), the OIM may take longer to respond to a Configuration

Management requested device search than the ORB default message timeout value. Increasing the ORBTCPReadTimeouts setting will ensure that Configuration Management client receives the proper data from an initiated device search, and not a time-out notification.

Action: edit the Configuration Management server start-up file (startWebLogic.sh), typically found at {$BEA_HOME}/user_projects/domains/CMDomain/bin/ and change the line

JAVA_OPTIONS="${SAVE_JAVAJ_OPTIONS}"

to

JAVA_OPTIONS="${SAVE_JAVAJ_OPTIONS}
-Dcom.sun.CORBA.transport.ORBTCPReadTimeouts=1:15000:300:1"

The Configuration Management server must be restarted after this file has been edited in order for the change to take effect.

Avoiding Errors When Disabling and Re-enabling CTM

The action of disabling CTM happens automatically. It is not done manually by issuing commands on cartridges, such as Network Processor managed devices. When CTM is re-enabled, commands are sent back to the device. On some devices and configurations, these are accepted without warnings being raised. For example, on Cisco IOS devices, you can re-enable CTM without any issues. On some device types, including Huawei, the device may report configuration errors when you try to re-enable CTM and existing configuration is resent to it.

To avoid such errors a cleanup script must be run as often as possible.

Additionally, do the following to ensure error free re-enabling of CTM:

- Ensure a CTM server cleanup of generic policies after successful installation.
- Optimize the interaction with the router. If the command set is empty (when deleting a configlet generic policy that only performs a **login, conf t, exit**), then the Network Processor will not perform the device login.

Tuning the Network Processor

The following sections provide information on how to increase Network Processor performance and stability.

Setting optimizeFirstCommit

Impacts: Improved Network Processor restart times.

Reasoning: By default, the Network Processor checks the status and performs configuration work on every device assigned to a particular Network Processor instance, every time the Network Processor is started or restarted. This work is performed even if there have been no changes to a device (as tracked by the Policy Server) since the last shutdown of the Network Processor.

This default behavior can be modified such that only devices that have been the target of service or network changes, since the Network Processor was brought down, will have the device configuration processing applied. This can reduce the start-up workload of the Network Processor by several orders of magnitude and hence reduce the time required to start or restart the Network Processor.

Action: Edit the Network Processor default.properties file (usually found at **/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/serviceactivator/networkprocessor/default.properties**). Change the optimizeFirstCommit processing flag from **false** to **true**.

> **Note:** Prior to IP Service Activator 5.2.3, the optimizeFirstCommit flag was called optimizeCommit.

Increasing the Allocated Memory of the Network Processor

Impacts: Improved Network Processor throughput.

Reasoning: The Network Processor loads and retains portions of device models in memory as it processes the devices it is assigned. The size of the allocated memory can be controlled by altering the size of the Java Virtual Machine (JVM) heap size that is established at Network Processor start-up. A larger heap will allow for more device models to be concurrently stored in memory and thus reduce the number of database fetches that Network Processor must request.

Action: Edit the networkprocessor initiation script (usually found at **/opt/OracleCommunications/ServiceActivator/bin/networkprocessor**). The default JVM heap size default setting is Xmx2048m (2 GB). The heap size should be large enough to allow for caching and buffering of the managed devices, but not so large as to cause excessive Java garbage collection. Additionally, this heap size should not be set to any value larger than the real memory capacity of the Network Processor host machine.

Increasing the CORBA Message Size for the Network Processor

This parameter is part of the default installation.

Impacts: Improved Network Processor Stability on large networks, when Configuration Management is deployed.

Reasoning: When Configuration Management is deployed, large messages are sometimes passed between the Policy Server and Network Processor relating to device lists and audit file lists. In order to insure that these messages can be sent in full, the parameter ORBgiopMaxMsgSize must be set to its highest level.

Action: Edit the Component Manager configuration script (**/opt/OracleCommunications/ServiceActivator/Config/cman.cfg**) and add the parameter and value *ORBgiopMaxMsgSize 4294967295* to the Network Processor startup command.

Limiting the Growth of Caches Within the Network Processor Buffer

Impacts: Increased device managing capacity of the Network Processor.

Reasoning: The Network Processor has a fixed amount of space, established as the JVM heap size, but some of the caches residing within the heap can be unbounded, or very large. It is important to set a limit on the size of the transaction queuing cache. So, in the event of receiving a very large transaction or group of transactions, the Network Processor and the Policy Server co-ordinate the transfer of the transactions to the Network Processor queue, rather than having the queue being boundless and overflow the JVM heap.

Action: Edit the Network Processor default.properties file (usually found at **/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/service**

activator/networkprocessor/default.properties). Add the parameter *cacheCapacity* processing flag and set the value to 20000000.

Increasing the Processor Thread Limit of the Network Processor

Impacts: Improved Network Processor throughput.

Reasoning: The Network Processor is a multi-threaded application, and as such can take advantage of available CPU-core resources on the host server. By default, a Network Processor instance is configured to utilize two CPU-cores. This can be expanded to utilize all available CPU-cores on the host server. Care must be taken not to configure a Network Processor instance to utilize more CPU-cores than are actually available, as this will impede performance.

Action: Edit the Network Processor default.properties file (usually found at **/opt/OracleCommunications/ServiceActivator/Config/networkProcessor/com/serviceactivator/networkprocessor/default.properties**). The processing flag to change is the *maxThreads* flag. Change this setting from *2* to 4 times the actual number of available CPU-cores. For example, if a Network Processor is deployed on a Sun T2000 (1x8 core CPU), and there are no other applications requiring resources on that host machine, the Oracle recommended maxThreads setting would be 32 (4 x 8 CPU-cores).

Additionally, adding the default.properties parameter, maxSchedulerThreads, and setting it to the same level improves Network Processor throughput.