Oracle® Communications IP Service Activator

Concepts

Release 7.4

E88200-01

December 2017

ORACLE®

Oracle Communications IP Service Activator Concepts, Release 7.4

E88200-01

Contents

3 Transactions

4 VPN and Connectivity Services

5 Policy-based Services

Preface

This guide provides an outline of the key features and benefits of Oracle Communications IP Service Activator, an overview of the distributed architecture, and an explanation of the basic concepts of VPN services and policy-based services.

Audience

This document is intended for sales and marketing, planners, and training (new users of IP Service Activator).

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

For more information, see the following documents in the Oracle Communications IP Service Activator documentation set:

- See *IP Service Activator Release Notes* for information related to this release of IP Service Activator.
- See *IP Service Activator Installation Guide* for system requirements and information on installing, upgrading and uninstalling IP Service Activator.
- See *IP Service Activator User's Guide* for information about setting up IP Service Activator and discovering and representing the network topology.
- See *IP Service Activator System Administrator's Guide* for information and procedures related the duties a system administrator performs in monitoring and managing IP Service Activator.

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.

1

Introduction

This chapter provides a high-level summary of Oracle Communications IP Service Activator. It describes some of the key features, and the architecture and components of IP Service Activator.

Overview of IP Service Activator

IP Service Activator defines and fully automates the activation of services on large-scale multi-vendor IP networks. IP Service Activator delivers end-to-end network control and gives you the flexibility to react in real time to new service and customer demands.

Using an intelligent, policy-based engine, IP Service Activator generates a detailed model of the managed network and the many features supported by the devices in that network. Based on these reported capabilities, you can then set up VPN-based services, manage Quality of Service (QoS), perform SLA monitoring, and apply access control measures in your network from one point of control.

IP Service Activator translates your service and policy definitions into the complex, device-specific configuration commands needed to implement the requested services. The Network Processor, working through service-specific cartridges, updates all relevant devices throughout the network in real time, making the implementation and day-to-day management of the network as simple as possible.

IP Service Activator helps you address the issues of delivering value-added services, such as IP VPNs, by providing an automated alternative to time-consuming, and potentially error-prone, manual processes typically employed for service activation.

Service Modules

IP Service Activator includes a set of service modules that model services and parameters, network-wide configuration relationships, policy-based activation logic, and service validation rules to implement services on your network. The service modules are vendor-agnostic. Additional device types are supported through cartridges to extend services to new types of vendor equipment.

The service modules comprise complex implementation details for the different services. However, the user interface interaction required to implement these services can remain simple and functionally oriented, making it easier for network operators to configure them. In other words, very complex tasks can be achieved with minimal operator interaction and no requirement for operators to retain detailed knowledge of the network implementation.

For more information, see "Configuration Template Module".

Flexible Multi-Vendor Support

IP Service Activator is designed as a multi-vendor solution, enabling you to deploy the most suitable devices for your service offerings without relying on equipment from a single vendor. For information about supported hardware, see *IP Service Activator Installation Guide*.

IP Service Activator's architecture is modular. Device management is handled through the Network Processor and corresponding cartridges, which are separate from the service activation and distribution components. This architecture allows new cartridges to be added as required. You can create custom cartridges using the IP Service Activator SDK.

The Network Processor is a data-driven XML-based processing engine that uses activation cartridges to support specific services on specific vendor equipment and operating system versions.

Requests for services and policies are converted into device and vendor-specific configuration commands, without the need to use templates and scripts. IP Service Activator determines which routers are affected by policies, the protocol to be used when updating device configurations, and the exact commands to be issued to the network. This frees network engineers to focus on other tasks.

Device Discovery and Management

IP Service Activator incorporates device discovery software to discover information about the physical network – routers, interfaces, and network segments – and create a detailed internal topology model. The discovery process also ascertains the capabilities of each device and interface. These capabilities define the services and policies that can be supported on each device.

The managed network can be displayed in the form of one or more topology maps, providing a logical and hierarchical representation of your network.

IP Service Activator keeps an internal model of the configuration of each device in the network, comprising the services and policies requested by users. IP Service Activator monitors each device to ensure that the expected configuration is not impacted by network events such as the device going down or by configuration changes made outside of IP Service Activator.

This removes the need for time-consuming manual interventions to track device capabilities and device configuration.

Policy-Based Management

IP Service Activator uses a flexible policy model, which means that users can apply policies to any point in the network (where supported by the device).

A role-based policy model provides flexibility in the way in which services and policies can be applied across the network. Device and interface roles enable you to logically group devices and interfaces (for example by customer or service package), so policies can be applied specifically to the targeted group.

Use of device roles and an inheritance model means that policies created at a central point can be applied to all relevant points in the network with a single action, eliminating the need for repetitive, error-prone manual processes.

The application of policy rules defining conditions and actions means that high-level policies can be defined by your organization's business requirements and the actual

details of implementation – the conversion from a high-level request to the actual configuration of a network device – are hidden from the operator.

Intelligent Data Modeling

The IP Service Activator knowledge store gives users a stateful understanding of network configuration as well as customer and service status. The knowledge store holds three types of inter-related information: a representation of network topology; details of services and policies to be configured; and full system data including status, fault, and logging information.

All changes made to the knowledge store are handled in the form of transactions, allowing you to exercise control over when and how changes take place. Used in conjunction with different levels of user access, transactions provide a granular and secure method for updating the knowledge store and configuring the network.

The intelligent data management within IP Service Activator automates the mapping of service changes to network device configurations. In addition, the data model can be shared with other operational support system (OSS) tools to support the deployment of additional new services.

OSS Integration Capabilities

The OSS Integration Manager (OIM) allows IP Service Activator to integrate with OSS applications, such as order entry, service assurance, fault management, and billing. The OIM consists of a CORBA-based API that allows third-party software to use IP Service Activator features without requiring detailed knowledge of the underlying implementation in IP Service Activator.

Oracle Communications has developed default support for applications from leading OSS providers.

IP Service Activator can be integrated with Oracle Communications Order and Service Management (OSM) using an optional Web Services component that is available during installation. For more information, see "Programmatic Intent-Based Network REST API" and "Web Service API".

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. For more information, see "Fault and Event Reporting".

Flexible Activation Extensibility

IP Service Activator includes multiple ways to extend its activation capabilities based on custom requirements. This includes extensions at the cartridge level as well as at user and API levels.

The Configuration Template Module (CTM) is a highly customizable configlet activation module that automates the creation of data-entry GUIs from simple XML configuration templates. The CTM provides automation for two purposes: the creation of configlets incorporating user or API parameter input, and the activation of routers. In this way, CTM replaces manual configurations and ensures configuration consistency for tasks of a repetitive nature.

You can use the CTM template in the IP Service Activator user interface (client) or by importing the CTM template into Design Studio. See "Configuration Template Module" for more information.

Scalability

IP Service Activator is designed to scale to carrier-class deployments, using its distributed software architecture. Additional servers can be added to increase its capacity. Servers can be installed geographically close to large groups of network elements, which can dramatically reduce the amount of management traffic running across the network.

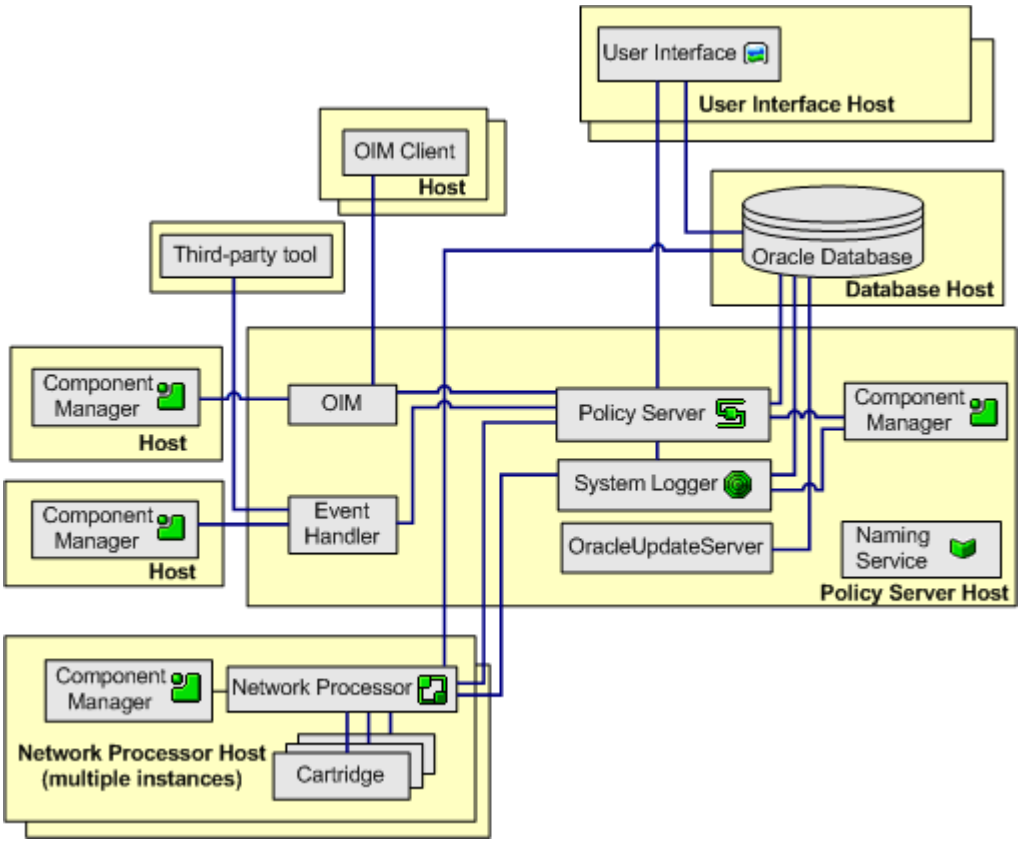
IPv6 Support

All interface management configuration policies and layer 3 Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) provisioning procedures, which are supported by the Cisco IOS cartridge, support both IPv4 and IPv6 addresses.

Distributed Architecture

The modular, distributed architecture of IP Service Activator is designed for scalability and resilience. It comprises a central policy server that coordinates the access of multiple user interfaces to a database and controls multiple distributed Network Processors. Each Network Processor uses one or more integrated vendor-specific cartridges to communicate to a number of network elements (devices). Figure 1–1 illustrates the IP Service Activator system architecture.

Figure 1–1 IP Service Activator System Architecture



Core Components

This section describes the core IP Service Activator components illustrated in the diagram above.

Policy Server and Database

The policy server is the central component of IP Service Activator. There is only one policy server for each IP Service Activator installation.

Functions of the policy server include the following:

- Validating data
- Managing user and system transactions
- Calculating all policies and services requested by user interfaces and third-party applications
- Discovering and modeling the physical network
- Coordinating access to the central database
- Downloading new configurations to the network processors for transmission to the devices.
- Core transaction monitoring functions

Two additional components are always installed with the policy server, the naming service and the system logger.

Naming Service

IP Service Activator uses CORBA (Common Object Request Broker Architecture) 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. As each component starts up, it registers with the naming service, passing the service its details. When one component needs to contact another, it contacts the naming service for details of the component.

System Logger

The system logger records system messages reported from IP Service Activator components or the managed network.

Database

Persistent storage of system data is maintained by a database, managed from the policy server. The database is typically located on a different host than the policy server. In live deployments, an Oracle 11gR2 database is recommended.

Network Processor and Cartridges

The Network Processor uses Activation Cartridges that include XML-based vendor-specific and service-specific definitions for a number of device types. Oracle Communications offers several cartridges that support a wide range of services across various vendors and operating systems.

The Network Processor component is also responsible for distributing configuration to devices. The integrated Network Processor-Cartridge architecture enables the Network Processor to manage a large range of device types.

Each cartridge is a software unit that provides configuration commands applicable to a family of vendor devices and operating systems, and a service (for example, QoS). Capabilities files apply to specific subsets of devices and operating systems in a vendor family.

The following are some common features of the cartridges:

- **Connection re-establishment**: Consistently retries to re-establish the network connection to the device in case of connection failure.
- **Save running configuration**: Provides option to save the running configuration to non-volatile storage after successful changes in the configuration.
- **SSH support**: Provides SSH connectivity to the device for secure sessions.
- **Password protection**: Encrypts the password for increased security, during configuration auditing process.
- **Enhanced VRF management**: This common VRF function allows you to set the wait time for the VRFs, number of retries; and enables you to choose whether the VRFs are to be deleted or saved in the device.
- **Threshold**: The cartridges provide consistent support for different thresholds, enabling you to monitor the network events regularly and configure alarms.
- **Failed associations**: This feature identifies the failed associations. You can flawlessly supervise and quarantine the failed associations in a configuration, minimizing the possibility of an error.
- **Offline Maintenance mode**: This feature allows you to update concretes and persisted device models, with no commands actually delivered to the device. You

can access Offline Maintenance mode in the IP Service Activator client or by using the OIM or OJDL.

- **Anonymous Login**: Supports anonymous login to the device.

Programmatic Intent-Based Network REST API

You can select the optional web services component when you install IP Service Activator. This allows you to use the programmatic intent-based network Representational State Transfer (REST) protocol over the web service API. You can use the REST web services API to provision customer-defined services and to integrate IP Service Activator with Oracle Communications Order and Service Management (OSM).

Using customized Groovy scripts with common REST methods, you can use calls through the REST web service to retrieve, update, delete, and create resources. For information about using the REST API, see *IP Service Activator API Developer's Guide*.

Web Service API

Web services is an optional component that provides a web service interface through which Oracle Communications Order and Service Management (OSM) can manage service activation activities and operations. For more information about OSM, see the Order and Service Management documentation. The interface is defined in the IP Service Activator Web Service Definition Language (WSDL) file. For information about installing and configuring web services with IP Service Activator, see *IP Service Activator Installation Guide* and *IP Service Activator API Developer's Guide*.

Oracle WebLogic Server is a required component for web services. For information about WebLogic Server, see the WebLogic Server documentation.

IP Service Activator transactions can be transparently handled by the web service. Each request sent to the web service contains a list of commands. The commands are then performed using a single transaction. The web service monitors the transactions and provides status notifications based on the result. See "Transactions" for more information about transactions.

Component Manager

The component manager is required on all host servers on which one or more IP Service Activator components are installed (except the user interface). The component manager is responsible for starting all components, monitoring and reporting their status, restarting any components that fail, and managing an orderly shutdown.

User Interfaces

The IP Service Activator graphical user interface (client) is used to display the current view of the managed network and to set up and apply new services and policies. It can represent multiple managed networks in both hierarchical and graphical form. It allows you to set up and apply policy and create VPNs using straightforward drag and drop operations. It reports on the status of applied configuration, the managed network, and IP Service Activator components, and it reports faults and events occurring throughout the network.

The client is a distributed multi-user program, enabling users logged on to different hosts to control IP Service Activator and maintain the database. The policy server coordinates the information between these components, ensuring that each user's view of IP Service Activator remains consistent.

Each client host maintains a local version of the object model, which reflects locally made changes.

The permissions defined for a client determine what is displayed on the client; where access to a class of objects is denied, the objects do not appear.

Configuration Template Module

The IP Service Activator Configuration Template module (CTM) allows you to do the following:

- Create, update, and delete configuration templates. To create a template means to define XML code that auto-generates data-entry dialog boxes and commands for configuring specific attributes on specific device or interface types.
- Activate templates by entering field data and sending the resulting command set to devices or interfaces.
- Perform related activities such as viewing template events, setting the lifetime of events, managing user access to CTM, and matching templates to schema versions.
- Import CTM templates using Oracle Communications Design Studio.

For more information, see "About the Configuration Template Module".

2

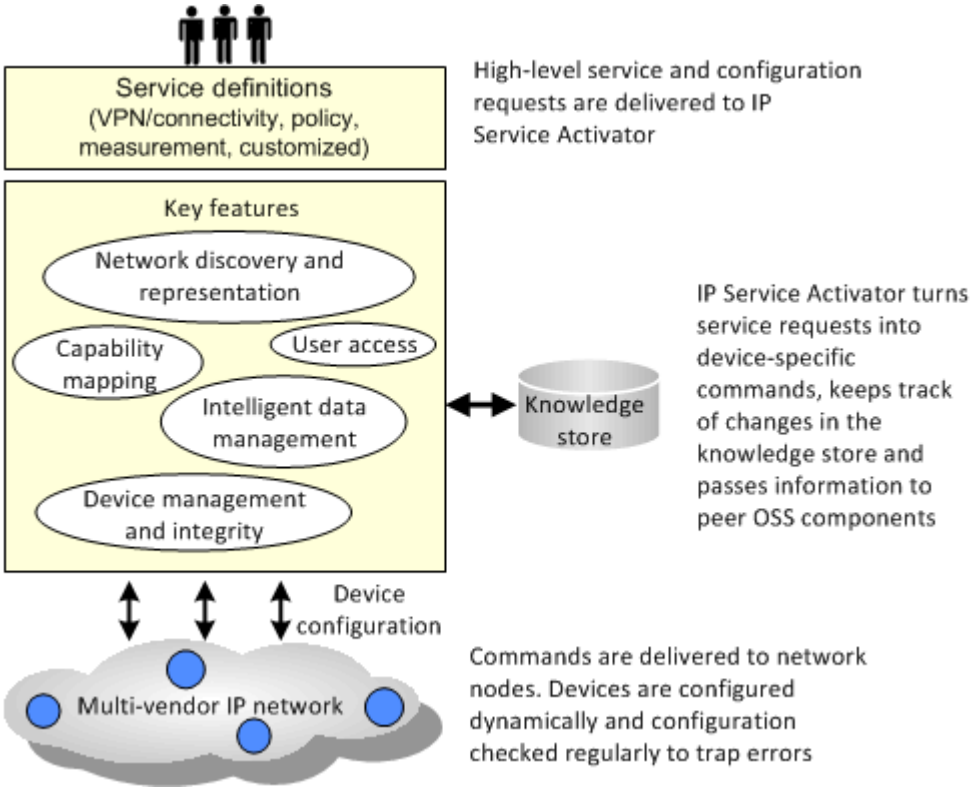
IP Service Activator Features

This chapter describes the key features of Oracle Communications IP Service Activator.

Figure 2–1 illustrates the core IP Service Activator functions and features.

These aspects are discussed further in the following sections.

Figure 2–1 Core IP Service Activator Functions and Features



Network Discovery and Representation

IP Service Activator incorporates a network discovery engine that is able to discover information about the physical network, including routers, interfaces and network segments, and create a detailed internal model.

The discovery process includes the following stages:

- Discovering devices, segments, and hosts

- Assigning devices to the components that are responsible for managing them
- Assigning policy roles to devices and interfaces
- Setting up the security and authentication parameters that IP Service Activator needs to configure devices
- Ascertaining the capabilities of devices and interfaces – the VPN, QoS and measurement options that are available at each network node

The Discovery Process

Given a valid IP address or DNS name of a device in the network, IP Service Activator can retrieve details of the device, its interfaces, and any sub-interfaces and ATM or Frame Relay virtual circuit (VC) endpoints present. Connectivity information is also obtained from querying the routing table.

It is possible to enter a number of IP addresses or a subnet mask and discover multiple devices at once. In addition, if public IP addresses are used, it is possible to search for connected devices by specifying the number of hops to go from the initial device. In this way, it is possible to discover the entire network in one pass.

Note: You can use a subnet mask only for discovery using IPv4, not for discovery using IPv6.

If any changes are made to the network configuration at any time, the discovery process can be re-run. You can either update the entire domain, or rediscover a single device.

The discovery process can also be set up to run automatically on a regular basis to ensure that the network topology is kept up-to-date. For example, you can configure IP Service Activator to run device discovery as an overnight process.

Access and Authentication

For each device, you must specify security parameters, such as user IDs and passwords, to allow IP Service Activator to configure the device. This information needs be set up only once.

The method used to communicate with the device depends on the options supported by the vendor and the level of security implemented in your network. You can use the SSH (Secure Shell) protocol to ensure secure communication between cartridges and routers. Control access by using passwords or an RSA key file. On Cisco devices, IP Service Activator supports the use of a TACACS+ server for authenticating and authorizing access to devices.

If standard security information is set up prior to discovery it will be applied automatically to all discovered devices. This can save time if you use the same access methods and passwords for a number of devices in your network.

Device and Interface Capabilities

At the end of the discovery process, IP Service Activator determines the hardware and software versions of the devices and then sets the capabilities of each discovered device accordingly.

Capabilities are set for each device and its interfaces, sub-interfaces, and VC endpoints to precisely define the VPN, QoS, access control, and measurement options available.

For interfaces and sub-interfaces, inbound and outbound capabilities are reported separately.

The Topology Model

After discovery is complete, IP Service Activator updates its representation of the topology model in the Knowledge Store:

- **Networks**: A network is a logical object only, representing all or part of the network to be managed. For each domain, a root-level network object is automatically created, which equates directly to the domain.

  To simplify the management of large and complex networks, you can set up subsidiary networks, each with its own topology map. Depending on the complexity of the network, you can create a multi-level hierarchy of subsidiary networks.

- **Devices**: Within IP Service Activator, a device can be defined as a network node that forwards IP packets, that is, a router or Layer 3 switch. Each device in the network is classified according to vendor type, which defines the Network Processor cartridge that will be used to manage the device. Note that it is possible to discover devices that cannot be managed.

  It is possible to create a virtual device manually in the client to represent a device that IP Service Activator has not discovered. This means that devices not managed by IP Service Activator, such as those in the core network or in the customer's domain, can be modeled.

- **Interfaces and sub-interfaces**: Interface and sub-interface objects represent the physical and logical interfaces on a device. This information is obtained from the interface table of the device during device discovery and the corresponding objects are created automatically.

- **PVCs**: For ATM and Frame Relay interfaces, IP Service Activator looks for PVCs (Permanent Virtual Circuits) on the device and creates VC endpoints to represent them. These endpoints can be connected in the user interface to create a representation of the virtual circuit.

- **Segments**: A segment object represents a locally-connected segment on an interface. Segments are classified according to type, for example, serial, bus or ATM cloud.

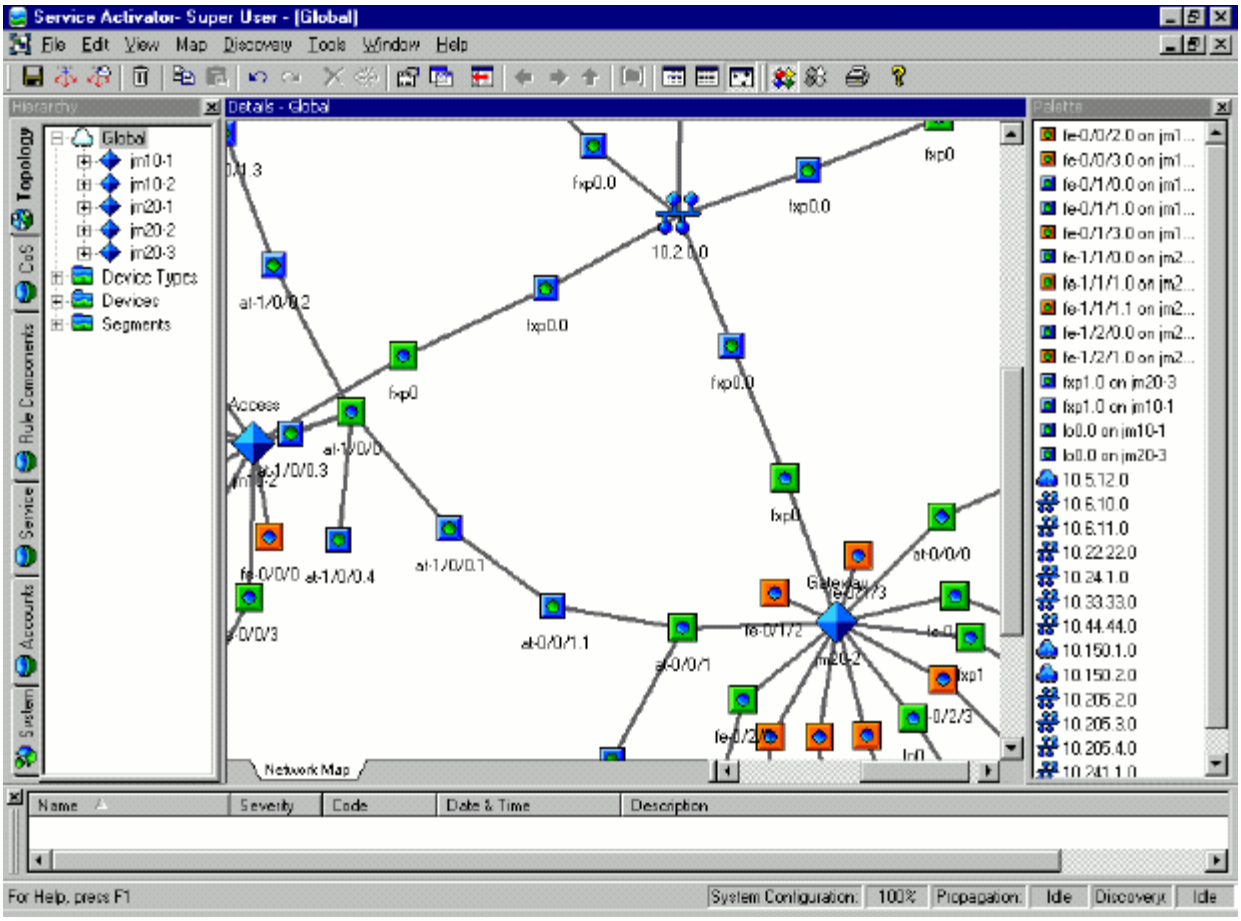
- **Hosts**: A host can be defined as a network node that does not forward IP packets, for example, a PC, workstation, server, or network printer. Hosts are found during the discovery process, and can be included for completeness although IP Service Activator does not manage them.

The hierarchical representation of the network topology appears in the Hierarchy pane of the client.

Mapping the Network

The managed network is displayed in the form of one or more topology maps in the client. As illustrated, the client provides a graphical and hierarchical representation of the managed network.

Figure 2–2 Hierarchical and Graphical Representation of the Network in the IP Service Activator Client



Map Views

The options for producing maps are flexible, and each network can have a number of alternative map views. For example, you can have one map showing the entire network and additional maps illustrating different portions of the network, such as the specific geographical or management regions. Alternatively, one map could display the network devices and segments and another could display the device interfaces and sub-interfaces. VPNs and their connected sites can also be mapped.

If the network is subdivided into two or more network objects, you can create maps for each sub-network, and drag and drop objects between maps.

The scale of the map can be changed to enable you to see the entire map at once or zoom into a particular section in more detail.

Manual Mapping

Network topology maps can be set up manually, by dragging and dropping network objects from an on-screen palette. You can control the positioning of objects by snapping them to an invisible grid layout. Connections between objects appear automatically.

The objects that appear in the palette are relevant to the object currently selected. For example if you choose a device, those interfaces that are not currently mapped are listed in the palette. If you select an interface, unmapped sub-interfaces and segments

are listed. This context-sensitivity can be turned off, in which case it is possible to select the types of objects to be listed.

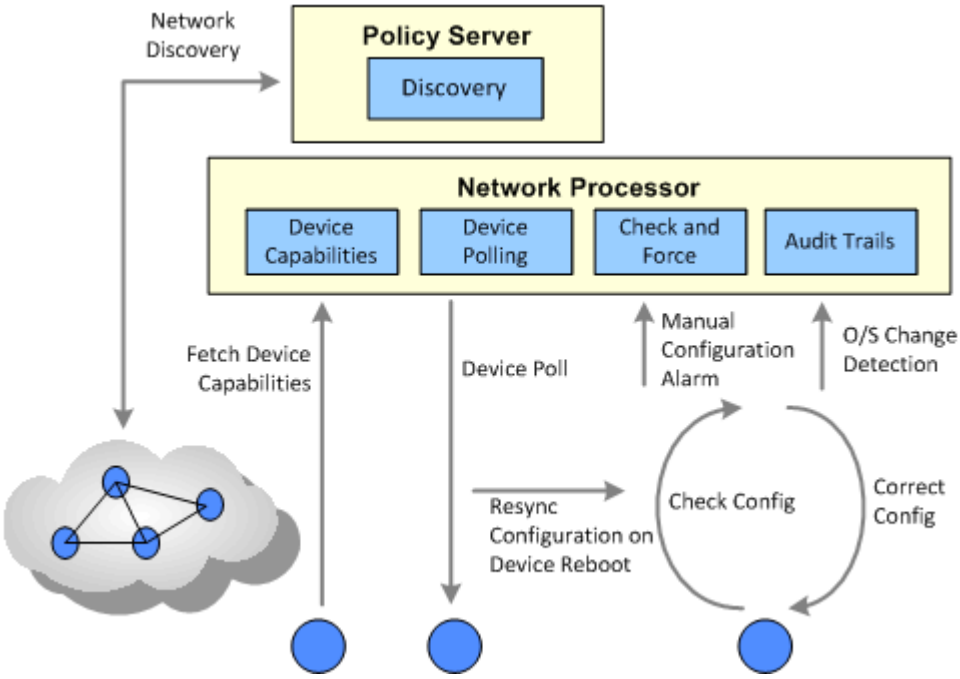
Automatic Mapping

It is also possible to apply an automatic layout option to the map. This arranges selected network objects in a logical manner. If automatic layout is selected before discovery, objects are automatically added to the map as they are discovered.

Device Management and Integrity

Figure 2–3 shows how IP Service Activator ensures device integrity.

Figure 2–3 Techniques for IP Service Activator Device Integrity



Communicating with the Device

The cartridges and Network Processor are the IP Service Activator components that are responsible for configuring individual network elements. They convert the abstract expressions of services, which have been calculated and validated by the central server, into the commands required to configure each element.

The device configuration is updated by issuing commands via the command-line interface (CLI). Access can be authenticated in a number of ways, including as a named or anonymous user, through a TACACS+ server or through the SSH protocol.

IP Service Activator maintains the authentication information for all devices. If the same access and authentication methods are used throughout the network, the parameters can be set up before device discovery and are then applied to all devices. Where devices have unique passwords, they need to have their security parameters set up separately.

Modeling Device Configuration

Rather than simply pushing commands out to each router, the cartridge or Network Processor manages the configuration intelligently, ensuring that the required configuration is not affected by the device going down or someone else making changes to the configuration. It does this by using an internal model of the configuration of each managed device, comprising the services and policies that have been requested by users.

The cartridge extracts the real configuration from the device and compares this with its internal model. If there are differences in the specific parts of the configuration that IP Service Activator configures, the configuration is updated. This comparison and update process is run every time the virtual configuration changes; that is, whenever any changes are made to requested services. If there is a mismatch, the configuration is updated.

The cartridges only ever reconfigure those interfaces that are specifically managed by IP Service Activator. If the requested configuration would conflict with essential routing and set-up configuration, it is not applied.

The Network Processor calculates the new device configuration based on the current transaction and the last pushed state for the device as stored in persisted data.

Ensuring Consistency and Integrity

IP Service Activator ensures that the configuration of each device is correctly maintained, even if the device goes down and configuration is lost.

Co-existence with Manual Configuration

You can set up IP Service Activator to co-exist with manually configured features, such as VRF tables and export maps.

Device Drivers can check for changes to device configuration that have been made manually by other users. Depending on a user-configurable option, a driver can force consistency of configuration automatically or issue a warning so that the device configuration can be investigated. The Network Processor does not force consistency.

Logging and Reporting

Each installed Network Processor records all configuration changes that it makes to devices under its control. The log files can be examined to check the date, time, user and details of each configuration change applied to each device.

Errors occurring in devices can be reported as SNMP traps, allowing integration with other network management tools.

Managing Data

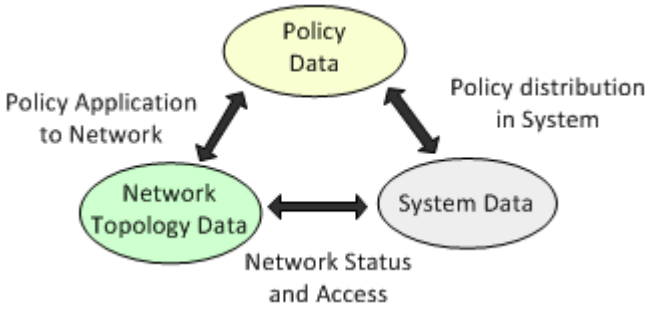
The way that IP Service Activator models and maintains data is fundamental to its operation.

The Knowledge Store

IP Service Activator's knowledge store is a repository of all required information. This information falls into three inter-related groups, as illustrated in Figure 2–4:

- **Network topology data**: A representation of the network being managed by IP Service Activator. This includes details of devices and their connectivity and the membership of VPNs.
- **Policy data**: Information relating to the policies and services that are used to configure the network.
- **System data**: Information about the IP Service Activator components and user details, including status, fault, and logging information.

Figure 2–4 Knowledge Store Information Groups



The knowledge store is also known as the object model: every element is modeled as an object instance of a pre-defined class. Each object has a set of attributes that identify it and hold its data, and associations that define its relationship to other objects.

The knowledge store is held in memory, but is mapped to a series of tables within the database for persistent storage. A simplified version of the knowledge store, known as the External Object Model (EOM), is available to external applications.

The External Object Model

The External Object Model (EOM) is a simplified version of IP Service Activator's knowledge store. It defines all the objects that can be accessed or updated by external applications, including their attributes and the relationships between them.

Using a subset of the entire object model means that user programs can create and access data objects without needing to know the underlying complexity of the entire object model. The External Object Model is independent of the command syntax – new attributes and objects can be added without requiring new commands.

Objects in the External Object Model are divided into three major categories:

- **Topology**: These objects represent the topology of the managed network, such as Network, Device and Interface objects.
- **Policy**: These objects represent elements used to define policies and services that can be configured on network nodes, such as Domain, Rule and Traffic Type objects.
- **System**: These objects represent IP Service Activator components and the event handler subscription model, such as Component, Subscription and Fault objects.

Transaction-based Processing

All changes made to the knowledge store are handled in the form of transactions. A transaction model allows users to exercise control over when and how changes take place. For example:

- A set of configuration changes can be created and grouped together and stored in the database for later application.
- Complex configuration changes can be broken down into a number of small transactions and applied incrementally.
- Specific configuration changes can be scheduled to be applied at a specified time in the future.
- A committed transaction may be rolled back to restore the device configuration to its previous state.

Defining Transactions

A transaction can consist of any sequence of operations, where an operation is one of the following:

- Creating or deleting an object
- Modifying an object's attributes
- Linking or unlinking two objects

For example, a single transaction could consist of setting up a number of customer sites and linking them together to form a VPN.

As well as transactions performed by users, actions performed by IP Service Activator are recorded as transactions.

Details of each committed transaction are archived, including the user who created it, its status, timing details, and the operations it comprises.

Timing of Transactions

Required changes can be broken down into a number of discrete transactions and implemented in a controlled manner. For example, a series of policy rules could be held in a number of transactions and their implementation phased over a period of time.

It is also possible to schedule a transaction so that it is applied at a specific date and time. For example, an operation to update the network could be applied at a time when traffic was at a minimum. At the appropriate time the transaction is automatically committed and the updates made.

Transaction Workflows

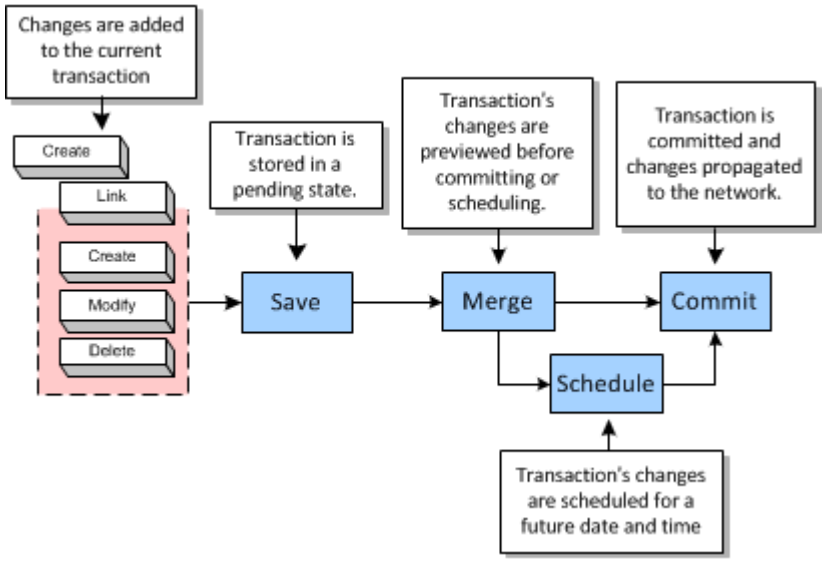
Depending on the security permissions set up for specific users, different individuals or groups of users can have different access rights for managing transactions. For example, some users may be able to create transactions and save them, while the right to commit or schedule existing transactions may be restricted to supervisory staff.

There are two potential workflows for working with transactions:

- One-stage commit model, where a user makes changes through the client and immediately commits the changes. For example, a super user might use this to set up additional users or basic data.
- Two-stage commit model, where one user saves the changes as a transaction and another merges the transaction to check its changes and finally commits it. For example, an operator might set up a VPN and a supervisor could check the changes before committing the transaction. In this model, the user can also save the transaction, and it can be scheduled to later point of time to commit the transaction.

This two stage process is shown in Figure 2–5.

Figure 2–5 Two-stage Transaction Commit Model



Rollback

A committed transaction can be rolled back in order to undo a set of changes that have been made. When a transaction is rolled back, its changes are removed from the object model and any configuration installed on network devices is removed. Any committed transaction can be rolled back providing that the changes involved are still valid within the current object model.

Network Representation

The client almost always reflects the network as it is currently configured. The only point at which it does not reflect the current state is when you are creating a transaction. Once the transaction is saved, the client reverts to the true state of the network.

Security Access

Access to IP Service Activator's powerful facilities for network configuration (through the client or from an external application through an API) can be controlled by extremely flexible security options. Security can be set up to control the following:

- Access to the client
- The operations a user can perform
- The information users can see through the client or OSS Integration Manager (OIM)
- User permissions on specific objects
- The ownership of individual object instances

User Authentication

Every user must have a login name and password to access IP Service Activator. To ensure security, system administrators can specify a minimum length for passwords,

set an expiry time for passwords and prevent password re-use. User accounts can be disabled after repeated log-in failure.

User and Group Permissions

Each IP Service Activator user is allocated to a user group, which defines the functions the user can perform. Each user group can have one of the following access types:

- **Read Only**: Group members can view objects but not modify them.
- **Read Write**: Group members may be able to view or modify objects.
- **Super User**: Group members can view and modify all objects.

For Read Write groups, you can set permissions which specify exactly which elements of IP Service Activator its members can view and modify. This enables you to limit access according to the role that users perform, or the customer accounts they are managing. Permissions can be set for the following:

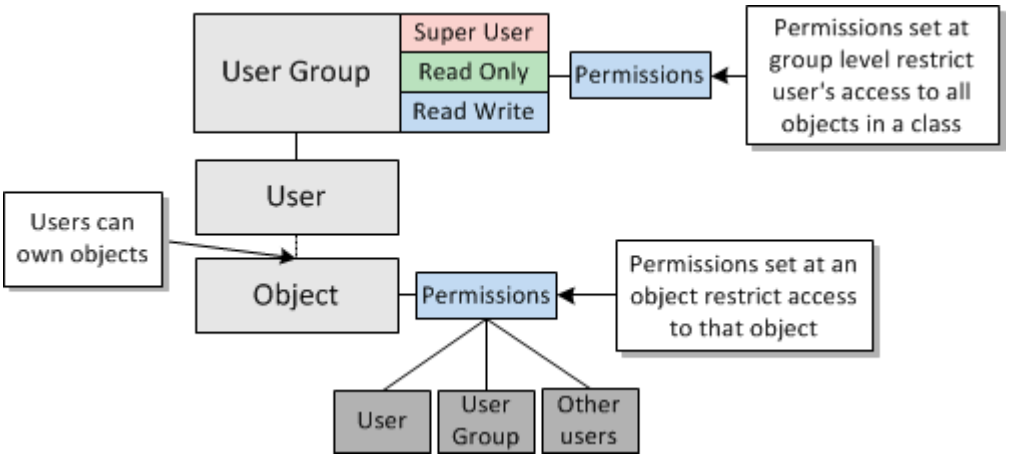
- Access permissions (such as Read-only, Modify, Create) can be set for specific object classes (for example, a user can be set to have Modify access on all VPNs)
- Specific operations (such as running a network discovery operation) can be permitted or denied

A user's access level can affect the appearance of the client. Objects that a user does not have permission to see do not appear.

Object Ownership and Permissions

For fine-grained control of security, it is possible to control access to specific object instances. Permissions can be set that apply to the owning user, the group to which the user belongs and all other users.

Figure 2–6 Object Ownership and Permission Levels



3

Transactions

Oracle Communications IP Service Activator uses a transaction-based model for updating the system and implementing configuration changes. This means that the changes you make through the client can be implemented immediately or saved in a pending state for future implementation. Configuration changes can be broken down into a number of transactions and implemented in a controlled manner.

About Transactions

A transaction is a set of changes made through the IP Service Activator client or using the OSS Integration Manager (OIM). These changes may include logical changes, such as creating a new domain, user group or users, as well as changes that affect device configuration, such as setting up a VPN or security policy.

As you make changes through the client, IP Service Activator adds those changes to a transaction referred to as the **current transaction**. You can choose when to stop the current transaction and how to handle it using the following options:

- Save the transaction in a pending state
- Commit the transaction's changes immediately and configure the network
- Schedule the transaction to be implemented at a specified date and time
- Discard or abort the transaction

These options support two methods for working with transactions. You can follow a one-stage update model, in which changes are made through the user interface and implemented immediately. Alternatively, you can follow a more secure two-stage update model, in which transactions are created and saved in a pending state, for checking and committing at a later date. You might want to use the two-stage update model in combination with user security options – allowing all users to create transactions and a subset of users to check and commit their work. For information about setting permissions on Read Write groups, see *IP Service Activator System Administrator's Guide*.

Whether you choose to follow a one or two-stage commit model, it may be possible to roll back a transaction's changes and remove associated configuration.

> **Note:** If other changes made in IP Service Activator are dependent on the modifications made by a transaction, you may be unable to perform a roll back.

IP Service Activator always attempts to reflect the current state of the network through the client. The client only differs from the network's current state when a user is in the

middle of the current transaction. As soon as he or she saves or schedules the transaction, the client reverts to the state of the network as it is currently configured. For more information, see "Local and Common Object Models".

Transaction Workflows

There are two potential workflows for transactions:

- The One-stage Commit Model, where a user makes some changes through the client and immediately commits the changes.
- The Two-stage Commit Model, where one user saves the changes as a transaction and another merges the transaction to check its changes and finally commits it.

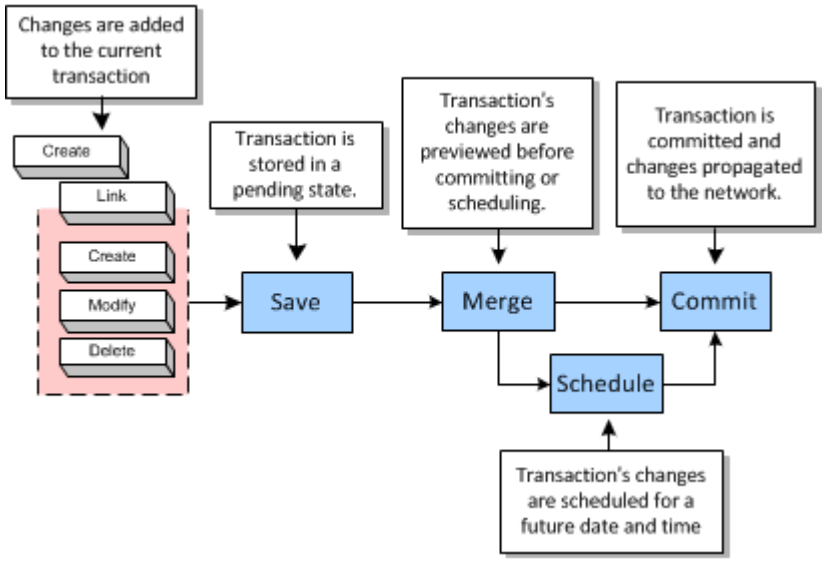
The One-stage Commit Model

In the one-stage commit model, changes made in the current transaction are committed immediately. This model is useful for initial system setup and demonstration purposes.

The Two-stage Commit Model

Figure 3–1 illustrates the two-stage commit model.

Figure 3–1 The Two-stage Commit Model



In this model, changes made through the client are added to the current transaction, which is saved and stored in a pending state. After saving, the transaction's changes must be checked by merging before it can be committed or scheduled for automatic commitment at a specified date and time. Merging performs a validation check and enables you to preview changes before committing them.

This model provides a granular and secure method for making updates and configuring the network. Changes can be broken down into a number of transactions and implemented in a controlled manner. For example, a series of policy rules can be held in a number of transactions and their implementation phased in over a period of time.

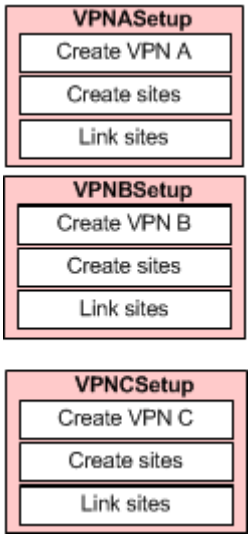
The action of creating and saving the transaction and committing the transaction may be performed by different users. In a distributed system, once a user has saved a transaction, other users working on remote clients can see the transaction and potentially implement it. Oracle therefore recommends that you give meaningful names to the transactions that you create.

To ensure that only trusted users can implement a transaction, you can set permissions on the actions associated with transactions. So, for example, you can allow all users to save transactions in a pending state but restrict the ability to implement transactions to a subset of users. For information about setting user access levels, see *IP Service Activator System Administrator's Guide*.

There are two options for creating and saving transactions:

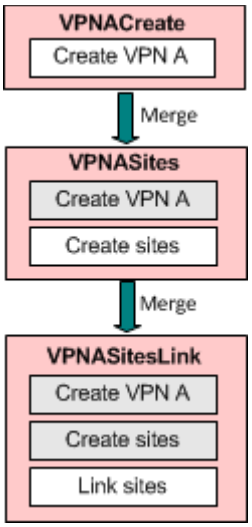
- Create discrete transactions with no overlap. In this option, each transaction is self-contained and performs distinct operations. For example, as illustrated in Figure 3–2, VPNASetup creates VPN A and the sites associated with it and links the sites with the VPN, VPNBSetup creates VPN B and the sites associated with it, and so on.

Figure 3–2 Discrete Transactions



- Create a transaction and save it in a pending state and then extend the transaction by merging it into the current transaction before performing additional tasks. This creates a series of transactions, where each transaction builds on the previous one. For example, as illustrated in Figure 3–3, the first transaction creates a VPN (VPNACreate), the second transaction merges VPNACreate and creates the sites associated with it (VPNASites), the third transaction merges VPNASites and links the sites with the VPN (VPNASitesLink), and so on.

Figure 3–3 Related Transactions



Oracle recommends that you adopt a naming convention when using the two-stage commit model, for example, by using the same prefix for related transactions.

Local and Common Object Models

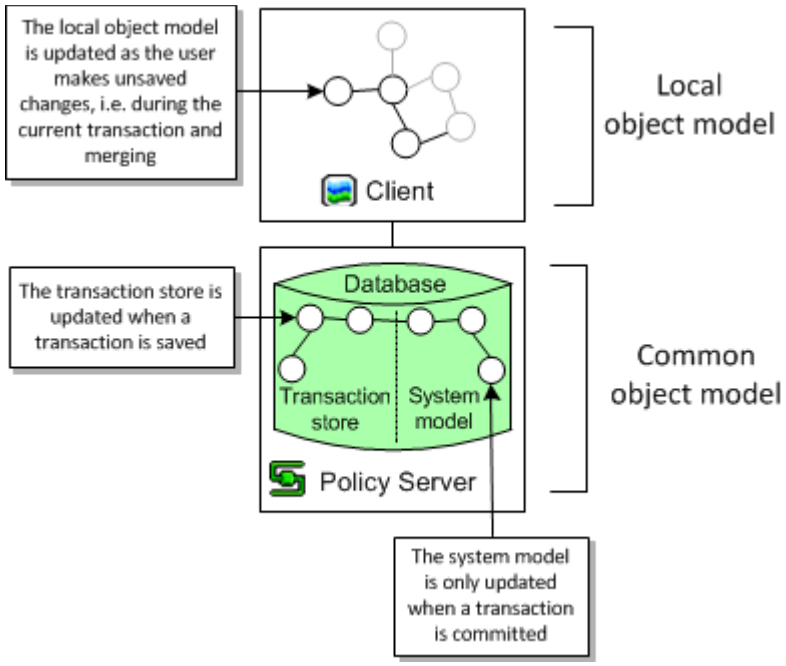
The changes made through the client and held in a transaction represent changes to IP Service Activator's object model. The common object model is maintained by the policy server and stored in the database, often located on the policy server host machine. The object model can be divided into two parts:

- The transaction store is updated when a transaction is saved. A new transaction object is created in the transaction store and is viewable by all clients. For information on the transaction store, see "The Transaction Store".
- The system model is updated with a transaction's changes when the transaction is committed.

As changes are made on a client host machine, IP Service Activator makes the changes locally. This document refers to this version of the object model as the local object model.

Figure 3–4 illustrates the relationship between the object models.

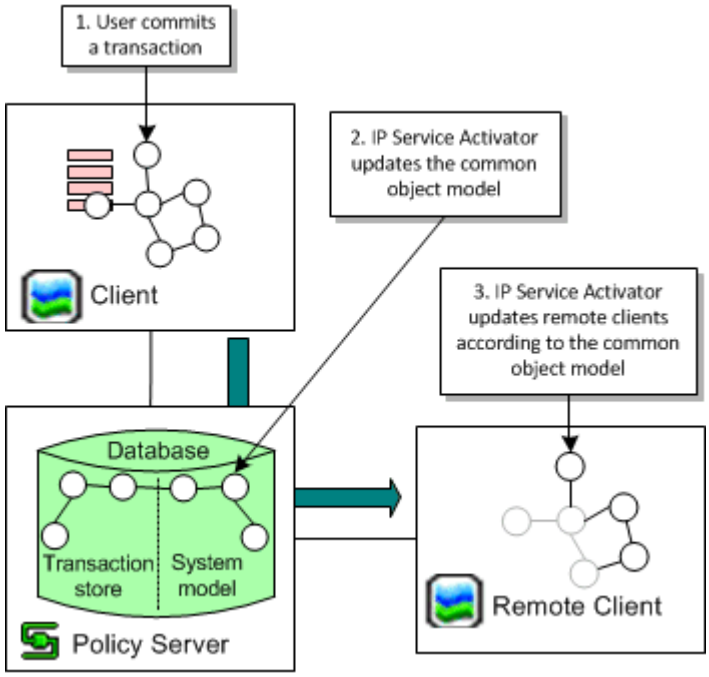
Figure 3–4 Local and Common Object Models



Each client's local object model is updated according to the common object model whenever a transaction is committed. The transaction may have been committed on the local host machine or on a remote client host.

Figure 3–5 illustrates the updating process that takes place when transactions are committed.

Figure 3–5 Updating the Object Models on Commit



When a transaction is committed, IP Service Activator compares the common object model to each user interface's local object model and resolves any inconsistencies by removing them from the local object model.

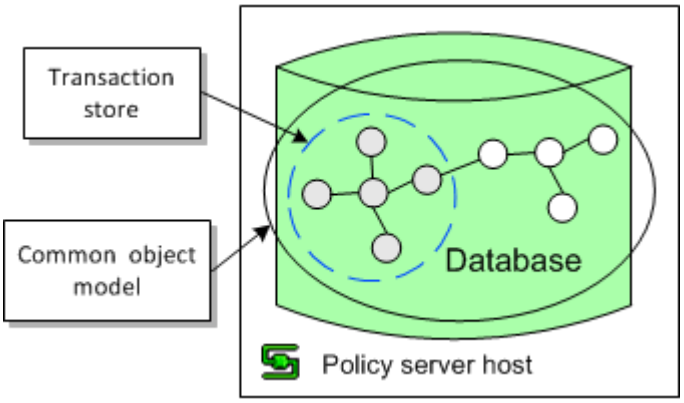
Note: A user with unsaved changes may lose those changes if they conflict with changes that have been committed to the common object model.

The Transaction Store

As part of the common object model, IP Service Activator maintains a transaction store which holds pending, scheduled and committed transactions. The store is updated whenever a transaction is saved or its status changes.

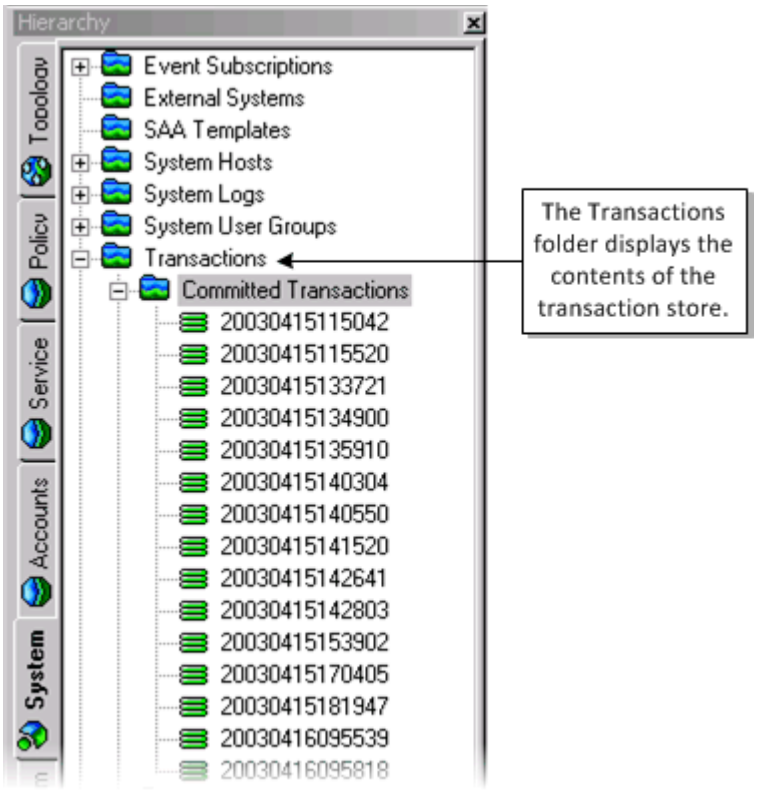
Figure 3–6 illustrates where the transaction store sits in the common object model within the Policy server host.

Figure 3–6 The Transaction Store



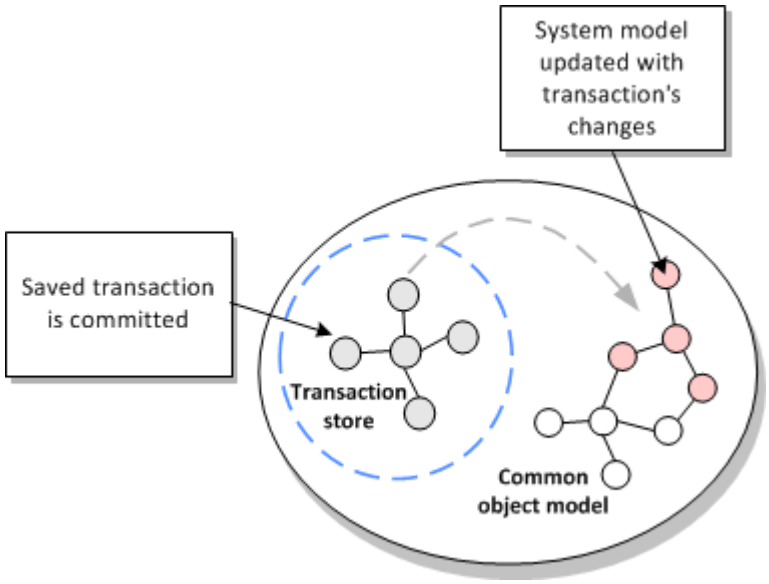
You can view the content of the transaction store on the System tab in the Transactions folder in the Hierarchy pane on the client, as shown in Figure 3–7.

Figure 3–7 The Transactions Folder



The system model section of the common object model is only updated with a transaction's changes when the transaction is committed, as illustrated in Figure 3–8.

Figure 3–8 Updating the System Model on Commit



Working with Transactions

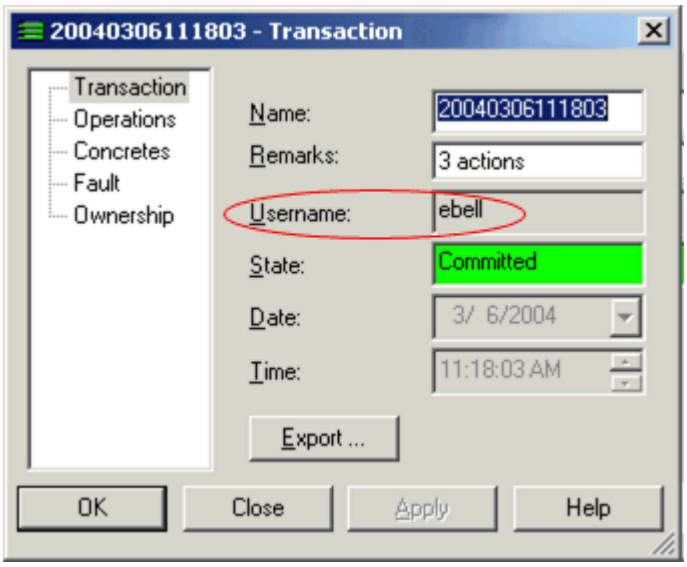
You can create, commit, save and schedule, merge and unmerge, and delete transactions. For more information about working with transactions, see *IP Service Activator User's Guide*.

Checking the Origin of a Transaction

There are three ways to check which user created a transaction. You can:

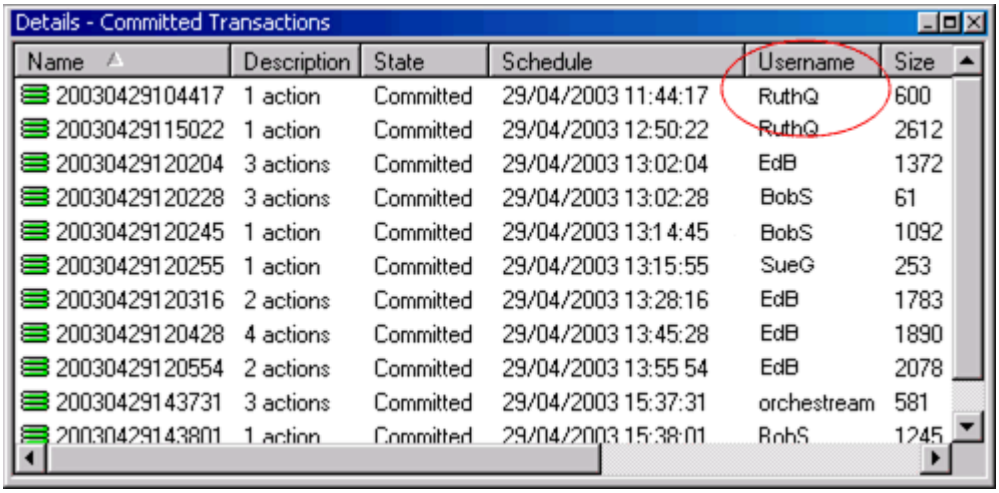
- Open the relevant transaction's properties dialog box and check the Username field on the Transaction property page, as shown in Figure 3–9.

Figure 3–9 The Username Field on the Transaction Property Page



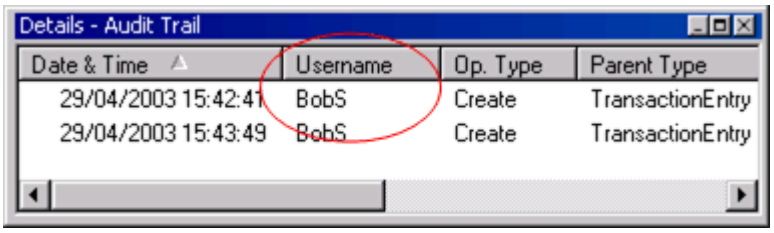
- List all transaction details in the Details pane of the Committed Transactions folder on the System tab, as shown in Figure 3–10.

Figure 3–10 The Username Column on the Details Pane



- List the audit trail details in the System Logs folder on the System tab, as shown in Figure 3–11.

Figure 3–11 The Username Column on the Audit Trail Log Details Pane



Selecting Transactions

You can select a single transaction or multiple transactions listed in the Details pane and select an action to perform on them. For example, you can merge a number of transactions in one step by multi-selecting them and selecting Merge from the pop-up menu. Where several transactions are selected, IP Service Activator processes them in the order in which they were selected.

Transaction Processing: Data Flow

When you commit changes in the IP Service Activator client and commit the transaction, several success or failure paths exist. This section describes some example paths.

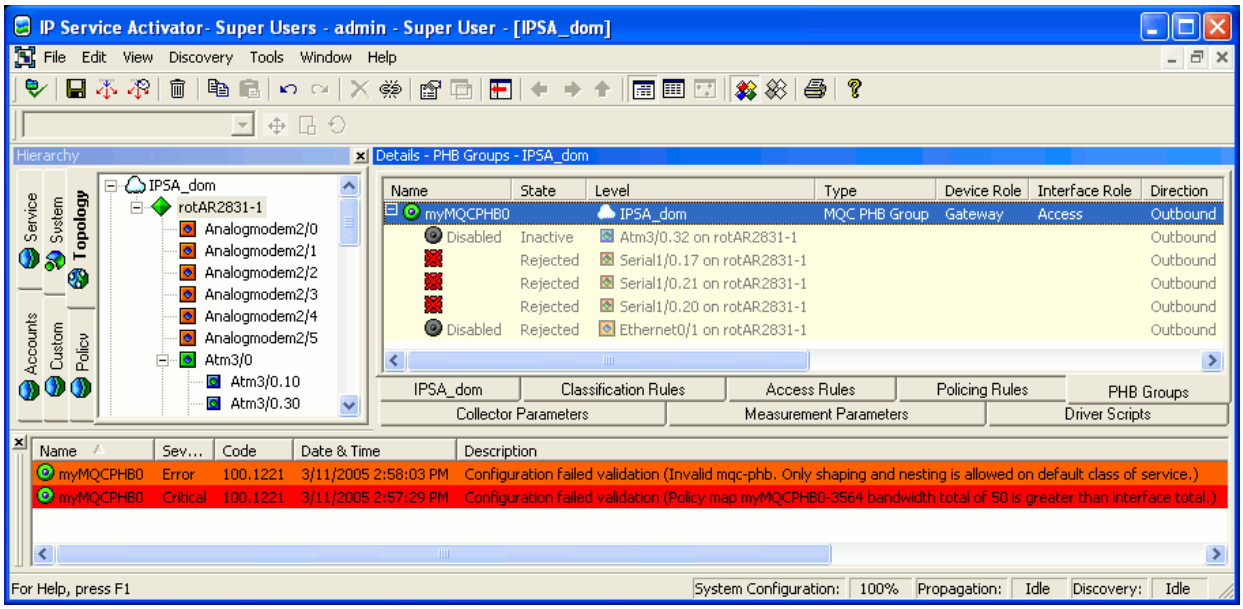
- When services are successfully activated by the Network Processor, their representations in the client (the Concretes) become “Installed”.
- When one or more concretes fail the Service Model (SM) or the Device Model (DM) validation rules, a fault (Critical or Error) is raised against the service and the invalid concretes affected by that transaction are rejected. The fault message includes “Configuration failed validation” and the validation rule that has been violated.

  For example, in Figure 3–12, two validation rules have been violated:

  – The MQC PHB includes shaping and other actions (for example, CBWFQ) at the same level. To correct this problem, a second MQC PHB including the other actions should be nested within the MQC PHB with shaping.

  – The MQC PHB includes queuing weight values that exceed the bandwidth of an interface. To correct this problem, either the queuing weight value should be reduced, the interface bandwidth increased, or if neither is possible, this MQC PHB should be disabled for this interface.

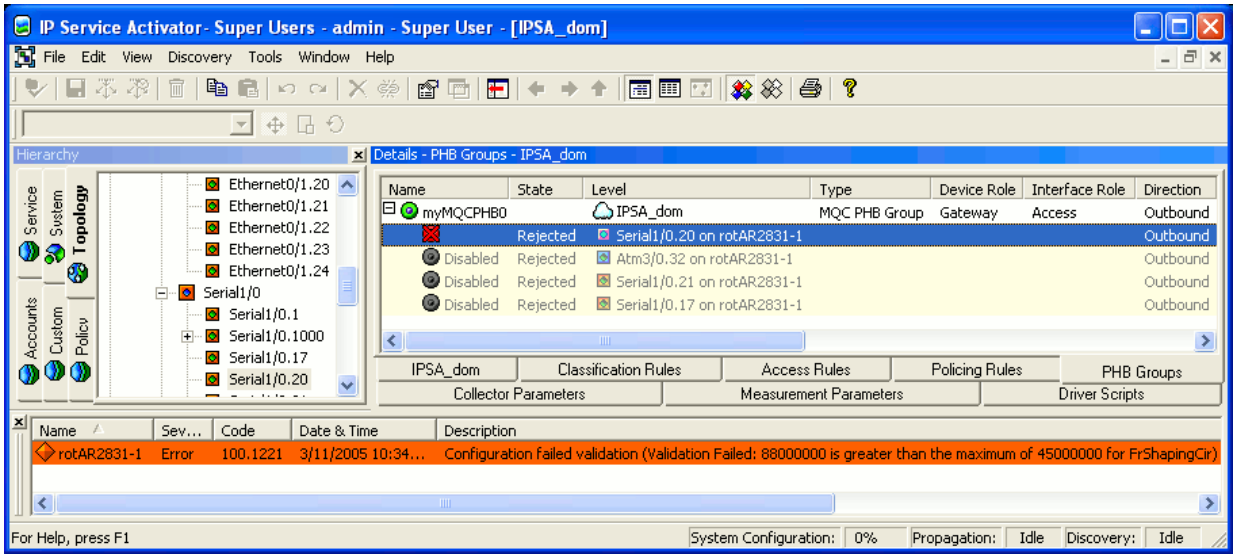
Figure 3–12 Validation Rule Faults



- When one or more concretes fail the DM Schema validation, a fault (Error) is raised against the device itself and all the concretes affected by that transaction are rejected. The fault message includes "Configuration failed validation (Validation failed ", the data element, and its current value.

  For example, in Figure 3–13, a schema validation rule has been violated. The MQC PHB includes traffic shaping with a CIR value of 88,000,000. The maximum value allowed for Traffic shaping CIR in a Frame Relay class is 45,000,000. To correct this problem, the CIR value should be reduced below the maximum value.

Figure 3–13 DM Schema Validation Fault



- After the Network Processor completes the data analysis and it is successful, it generates the commands, connects to the device, and starts changing the device configuration. The response from the device is analyzed after each command:

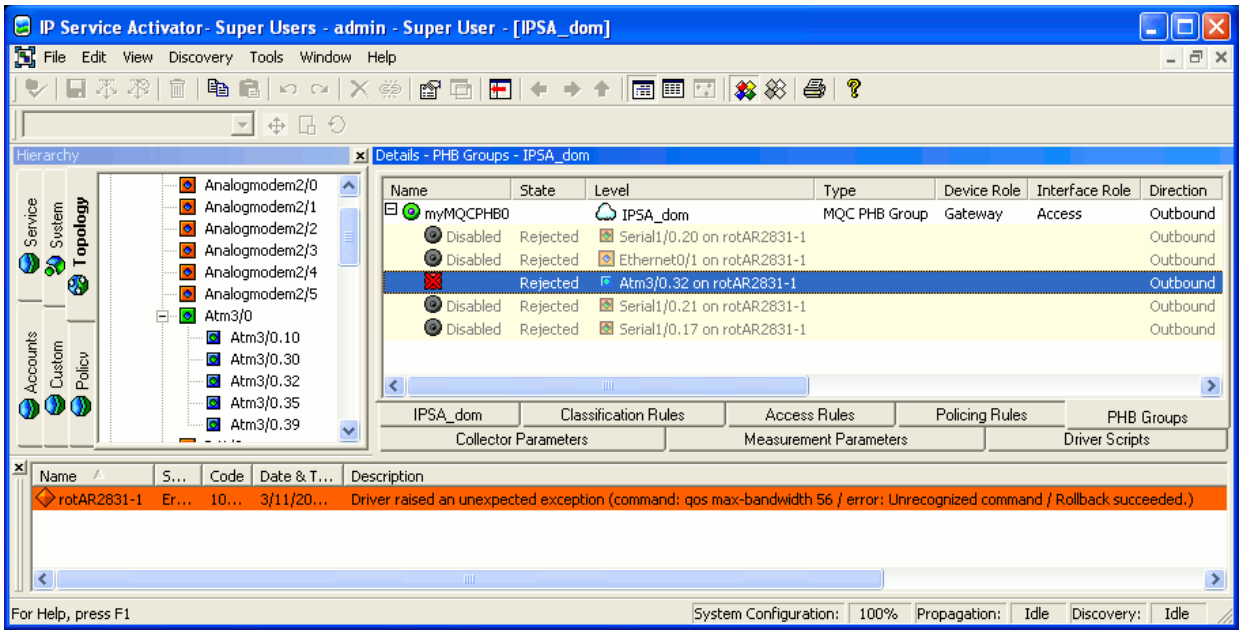
– **Success response**: If the device response includes a success message or no message at all (only a prompt) then the command is considered successful and the next one is issued. The success message list can be edited by the system administrator.

– **Error response**: If the response from the device matches one of the known error patterns, then a fault (Error) is raised against the device itself, the invalid concretes affected by that transaction are rejected and any failed configuration is rolled back. The known error patterns (regular expressions) and their associated messages are listed below:

```
<cmd:errorPattern>
    <cmd:pattern>(?s).*Unrecognized command.*</cmd:pattern>
    <cmd:message>Unrecognized command</cmd:message>
</cmd:errorPattern>
<cmd:errorPattern>
    <cmd:pattern>(?s).*Wrong parameter found.*</cmd:pattern>
    <cmd:message>Wrong parameter found</cmd:message>
</cmd:errorPattern>
<cmd:errorPattern>
    <cmd:pattern>(?s).*Not supported on .*</cmd:pattern>
    <cmd:message>Not supported</cmd:message>
</cmd:errorPattern>
<cmd:errorPattern>
    <cmd:pattern>(?s).*Cannot change interface.*</cmd:pattern>
    <cmd:message>Cannot change interface</cmd:message>
</cmd:errorPattern>
<cmd:errorPattern>
    <cmd:pattern>(?s).* Incomplete command.*</cmd:pattern>
    <cmd:message>Incomplete command</cmd:message>
</cmd:errorPattern>
<cmd:errorPattern>
    <cmd:pattern>(?s).* Too many parameters.*</cmd:pattern>
    <cmd:message>Too many parameters</cmd:message>
</cmd:errorPattern>
```

The fault message includes the rejected command and the message associated to the error pattern. For example, in Figure 3–14, the transaction has been canceled and a fault raised when the device responded with “Unrecognized command” to the “qos max-bandwidth 56" command. Rollback has succeeded. To correct this problem, an investigation is necessary. The **networkprocessor.log** and the **np***cartridgeName***.audit.log** should be analyzed. The problem may be that the interface does not support this service.

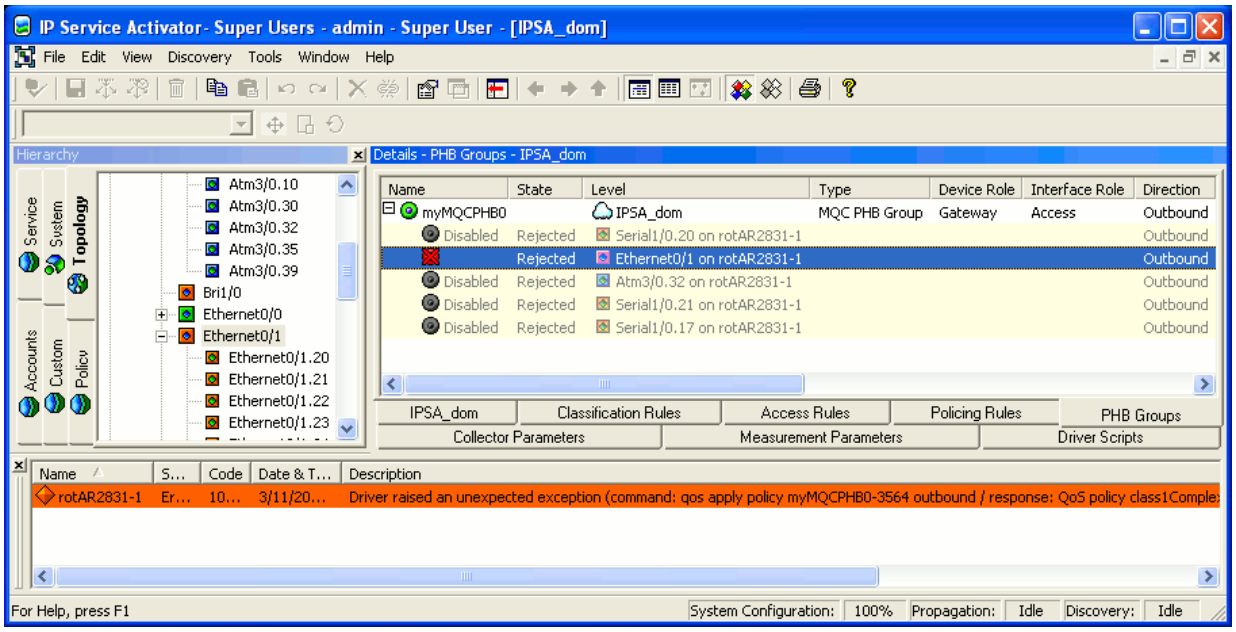
Figure 3–14 Error Response



– **Unknown error**: If the response from the device does not match any success or error pattern, then the response is considered an unknown error. A fault (Error) is raised against the device itself, the invalid concretes affected by that transaction are rejected, and any failed configuration is rolled back. The fault message includes the rejected command and the actual message returned by the device. Note that due to fault display constraints, successive white space and new line characters are removed from the device response.

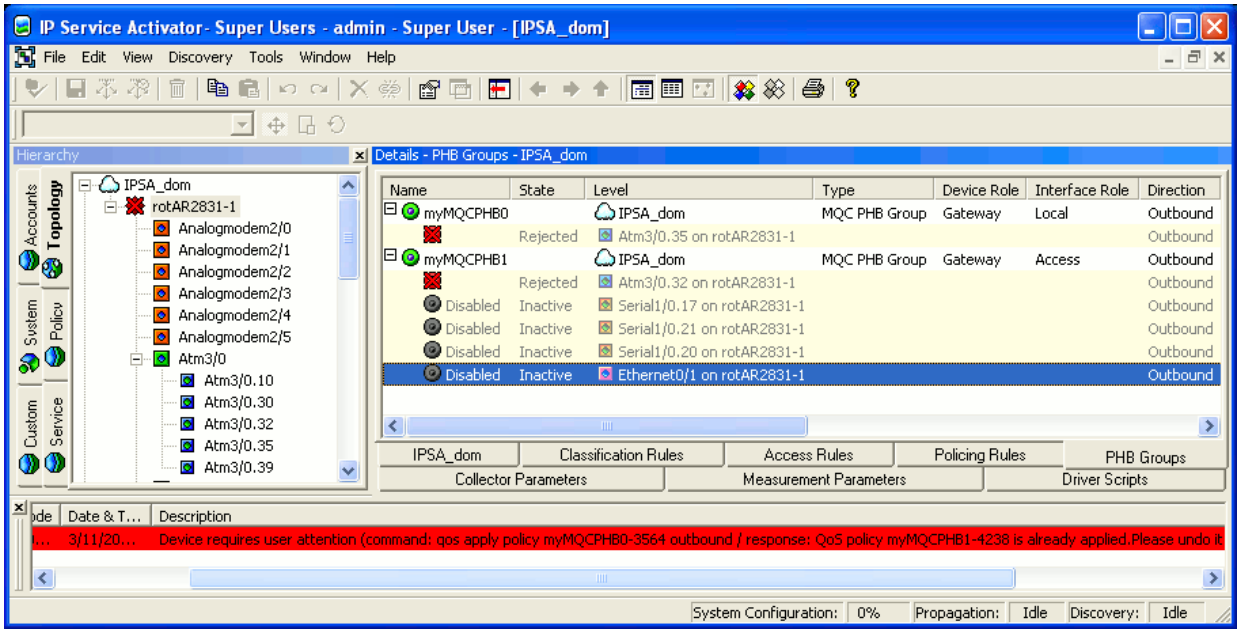
  For example, in Figure 3–15, the transaction has been canceled and a fault raised when the device responded with "QoS policy *name* is already applied" to the "qos apply policy *name* outbound" command. Rollback has succeeded. To correct this problem, an investigation is necessary. The **networkprocessor.log** and the **np***cartridgeName***.audit.log** should be analyzed. This problem may be due to a conflict with manually installed configuration. The conflict must be manually resolved.

Figure 3–15 Unknown Error Response



– **Rollback fails**: If the rollback fails, then the fault is "Critical", as shown in Figure 3–16. This problem may be due to a conflict with manually installed configuration. The conflict must be manually resolved.

Figure 3–16 Rollback Fail Response



The invalid concretes should be disabled until the problem is corrected. The valid concretes can be disabled and re-enabled in order to have them successfully installed. For more information, see *IP Service Activator System Administrator's Guide*.

Concretes

A concrete rule is an implementation of an abstract classification rule, policing rule or access rule that applies to a specific point in the network, i.e., at a particular interface. Each abstract rule set up can result in a number of concrete rules.

Each concrete rule has an associated status:

- **Inactive**: The rule has been created, but has not yet been propagated to the network processors.
- **Disabled**: The rule has been disabled by the user.
- **Active**: The rule has been propagated to a network processor but is not yet installed on a device (for example, because the date and time limitations do not apply yet).
- **Conflict**: The rule conflicts with an existing rule.
- **Installed**: The rule has been successfully installed on the designated device.
- **Rejected**: The network processor failed to install the rule; the rule has been discarded.
- **Uninstalled**: The rule has been disabled by the user and successfully removed from the device.
- **UninstallFailed**: The rule has been disabled by the user, but failed to be removed from the device. (A fault may be raised in the Faults pane.)

Searching for Concrete Objects

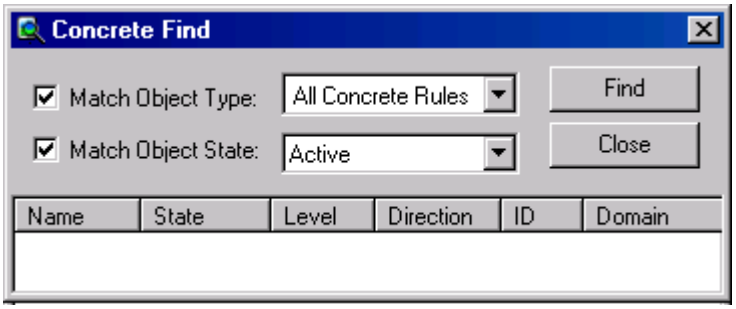
A concrete object search attempts to locate concrete objects that match the specified state. This means, for example, that you can search for all concrete access rules whose state is 'Installed' or search for VPNs whose state is 'Active'.

To search for concrete objects:

1. Do one of the following:
   - From the Tools menu, select **Concrete Find**.
   - In the Statistics Summary pane, double-click a cell.

     The Concrete Find dialog box appears, as in Figure 3–17.

Figure 3–17 Concrete Objects Search



2. Select the **Match Object Type** option and select a concrete object type from the drop-down menu.
3. (Optional) Select the **Match Object State** option and select a state from the drop-down menu.

The search is restricted by status.

4. Start the search by clicking **Find**.

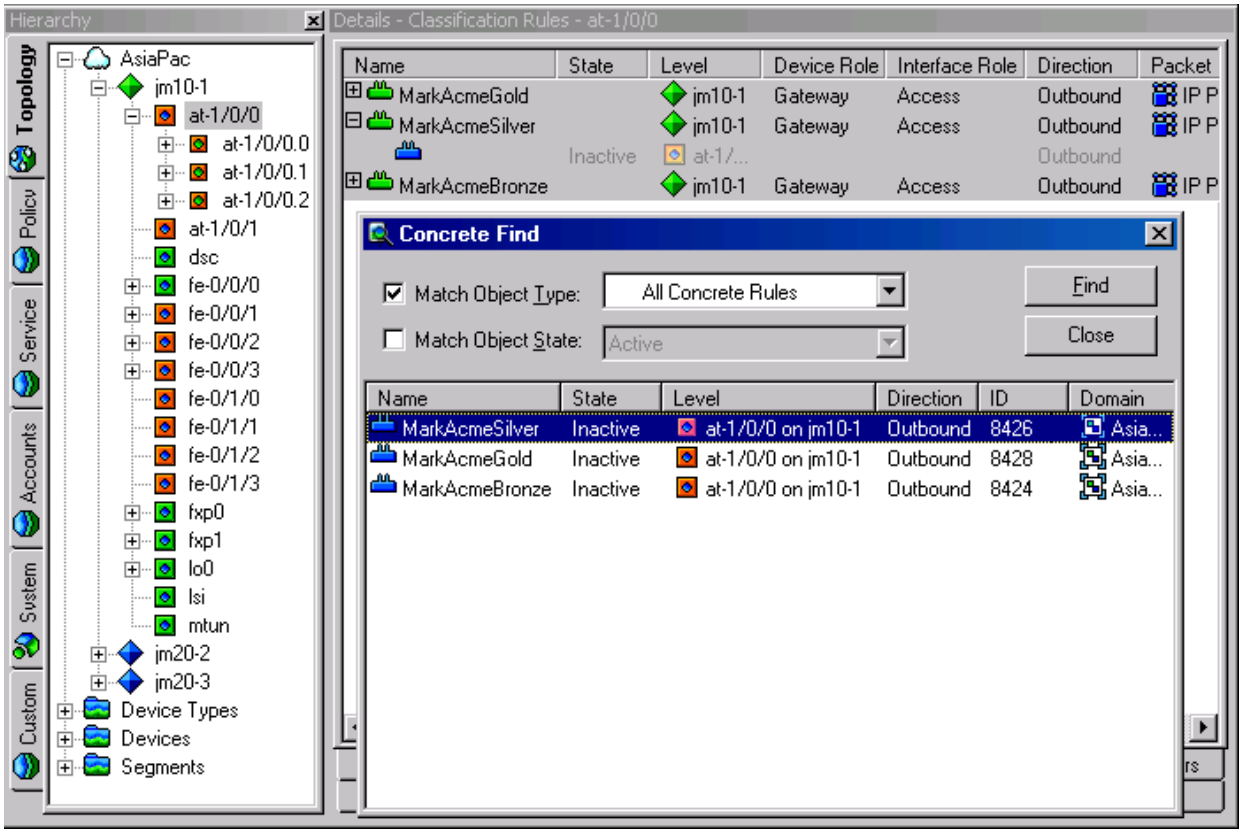
   IP Service Activator searches the object model for concrete objects that match the search criteria.

   For each found concrete policy element, the following details are listed:

   - **Name**: Name of the object
   - **State**: Status of the object
   - **Level**: Name of the topology object at which the concrete object applies
   - **Direction**: Whether the object is applied inbound or outbound to the interface
   - **ID**: IP Service Activator object ID
   - **Domain**: Name of the domain

5. Double-click a found concrete object to view the point at which it applies in the Hierarchy pane and the details of the rule in the Details pane, as shown in Figure 3–18.

Figure 3–18 Displaying Hierarchy and Rule Details for a Found Object



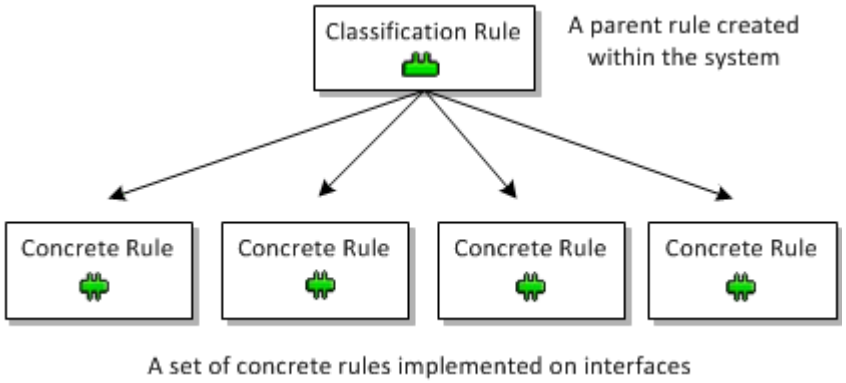
Abstract and Concrete Policy Elements

Committing a transaction that propagates a rule, PHB group, or driver script to the network might result in the creation of concrete policy elements. A concrete policy element is an implementation of a defined rule, PHB group or driver script that applies to a specific point in the network, that is, at a particular interface, subinterface

or VC endpoint. Each abstract or parent policy element set-up can result in a number of concrete policy elements.

Figure 3–19 illustrates the concept with a classification rule.

Figure 3–19 Abstract Classification Rule Resulting in Concrete Rules



You can list implemented rules, PHB groups, VPNs or driver scripts for a policy target in the Details pane. IP Service Activator indicates whether the policy element is abstract or concrete using colored backgrounds:

- **White background**: Abstract policy element that has been set up on the selected object
- **Yellow background, indented**: Concrete policy element that exists at the current or a lower level in the hierarchy
- **Gray background**: Abstract policy element that has been inherited from a higher level object
- **Gray background, indented**: Concrete policy element for an inherited policy element

Abstract and Concrete Rules

The term **abstract rule** refers to the rule object created by the user as opposed to an implementation of the rule as installed in the network. A **concrete rule** is an implementation of a created classification rule, policing rule or access rule that is installed at a specific point in the network, that is, at a particular interface, sub-interface or VC endpoint. Each abstract or parent rule set up can result in a number of concrete rules.

Rule Status

An abstract rule might have a status of **active** or **disabled**. The status of an abstract rule is indicated by the appearance of its icon in the Details pane. See the IP Service Activator online Help for abstract rule status icons.

A concrete rule is an implementation of an abstract classification rule, policing rule or access rule that applies to a specific point in the network, i.e. at a particular interface. Each abstract rule set up can result in a number of concrete rules.

A concrete rule may have the following status values:

- **Inactive**: The rule has been applied to an interface but has not yet been propagated to network processors.

- **Active**: The rule has been propagated to a network processor but is not installed on a device at present.
- **Conflict**: The rule conflicts with an existing rule or fails validation, for example, the action performed by a rule is not supported by an interface.
- **Rejected**: The network processor failed to install the rule and it has been discarded.
- **Installed**: The rule has been successfully installed on the designated device.

Concrete Activation Status

The concrete activation status allows flow-though applications of IP Service Activator to reliably track the progress of any IP Service Activator transaction.

The concrete activation status directly indicates whether a transaction's configuration changes, corresponding to each individual concrete, have been successfully activated on the network. The system maintains the concrete activation status for all client transactions including those committed or scheduled through the client and the OIM.

The activation status of a concrete object can be one of the following values:

- **Inactive**: Concrete is not yet submitted.
- **Active**: Activation is in progress. Passed indicates that the device is in sync with the current version of the service. **Not Available** indicates that no audit information is available.
- **Installed (Installed)**: The concrete is installed and is in sync with the current IP Service Activator model.
- **Installed (Installed)**: The concrete is installed, but no audit information is available.
- **Conflict**: The current version of the concrete is in conflict with system capabilities and validation rules. Passed indicates that the device is in sync with the previous version of the service. **Not Available** indicates that no audit information is available.
- **Rejected**: The current version of the concrete is in conflict with device validation rules. Passed indicates that the device is in sync with the previous version of the service. **Not Available** indicates that no audit information is available.
- **Disabled**: Concrete was removed from the device. The audit should not report any configuration for this service.
- **Mismatched (Active)**: Activation is in progress. The audit indicates that the device is out of sync with the current IP Service Activator model.
- **Mismatched (Conflict)**: The current version of the concrete is in conflict with system capabilities and validation rules. The audit indicates that the device is out of sync with the previous version of the service.
- **Mismatched (Rejected)**: The current version of the concrete is in conflict with device validation rules. The audit indicates that the device is out of sync with the previous version of the service.
- **Mismatched (Installed)**: The concrete is installed, but is out of sync with the current IP Service Activator model.

Turning Off and On the Concrete Audit State Feature

By default, this feature is turned off. You must be aware of the following if you are turning this feature on:

- When the concrete audit state feature is turned on, you must not perform any configuration changes affecting a device that is undergoing a device audit. Your changes can be lost if:
  - A device audit is in progress. (This results in an update of the Audit State of the device and its concretes.)
  - You make a change to the device configuration, or to any object within the device (interface, sub-interface, VC), or to any service object that affects concretes on the device.
- If the device audit updates the Audit State of the concretes before the user commits the configuration changes, the user sees a message that they need to reenter their changes.

The concrete audit state feature is described in detail in the IP Service Activator online Help.

To turn on the Concrete Audit State feature:

1. Open the **default.properties** file.
2. Set the value of suppressAuditFeedback to **False**.

To turn off the Concrete Audit State feature:

1. Open the **default.properties** file.
2. Set the value of suppressAuditFeedback to **True**.

Note: If the suppressAuditFeedback field does not appear in **default.properties**, add it manually. When the suppressAuditFeedback field does not appear, its value defaults to **True** (the feature is suppressed).

When a per-service audit is initiated with the suppressAuditFeedback property 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.

If Concretes are not Created as Expected

If no concrete PHB groups are listed for an abstract PHB group, check the following:

- The correct device and interface roles have been associated with the PHB group and assigned to the relevant policy targets.
- Devices to which the PHB group should apply are managed.

IP Service Activator applies only one concrete PHB group to an interface. An abstract PHB group that has been applied at a lower level in the hierarchy overrides any PHB groups that have been applied at a higher level. A concrete standard PHB group that configures FRTS or ATM traffic shaping can be installed on an interface that has a concrete MQC PHB group whose mechanisms do not conflict with the standard PHB group.

4

VPN and Connectivity Services

This chapter provides an overview of the VPN and network connectivity services supported by Oracle Communications IP Service Activator.

Layer 3 MPLS VPNs (RFC 4364)

IP Service Activator supports the established IETF standard (RFC 4364bis) for Layer 3 VPNs on Cisco and Juniper M-series. MPLS is used to forward packets over the backbone while BGP is used to distribute routes, offering a scalable alternative to fully meshed circuit or tunnel-based IP VPNs. Security and privacy within an MPLS VPN is achieved by limiting the distribution of routing information to members of the VPN.

Using IP Service Activator allows you to set up VPNs quickly and easily by defining the appropriate customer sites and specifying how they are linked together into VPNs. The relevant routers throughout the network are then configured automatically.

For complete details on MPLS VPN support for specific vendor devices, please refer to the appropriate cartridge guide.

Key Concepts

This section describes some of the key MPLS VPN concepts, as implemented in IP Service Activator.

MPLS

MPLS uses a technique called label switching to forward data throughout the network. The routers within an MPLS network that are responsible for label processing are known as Label Switched Routers (LSRs), and the path followed by data is known as a Label Switched Path (LSP).

On entry to an MPLS network, one or more fixed-format labels are inserted at the front of each packet. This label tells switching nodes throughout the network how to process and forward the data. The packet's path through the network is defined by its initial labeling – subsequent mapping of labels is defined by the initial label allocation.

Each LSR in the MPLS network normally runs a Label Distribution Protocol (LDP) that distributes the labels that are to be used to forward packets across the MPLS network using peer-to-peer negotiation from network edge to network edge. These labels are associated with Forwarding Equivalence Classes (FECs) that define an IP address prefix for an egress point from the network. The MPLS labels only have significance between these LSR-peers.

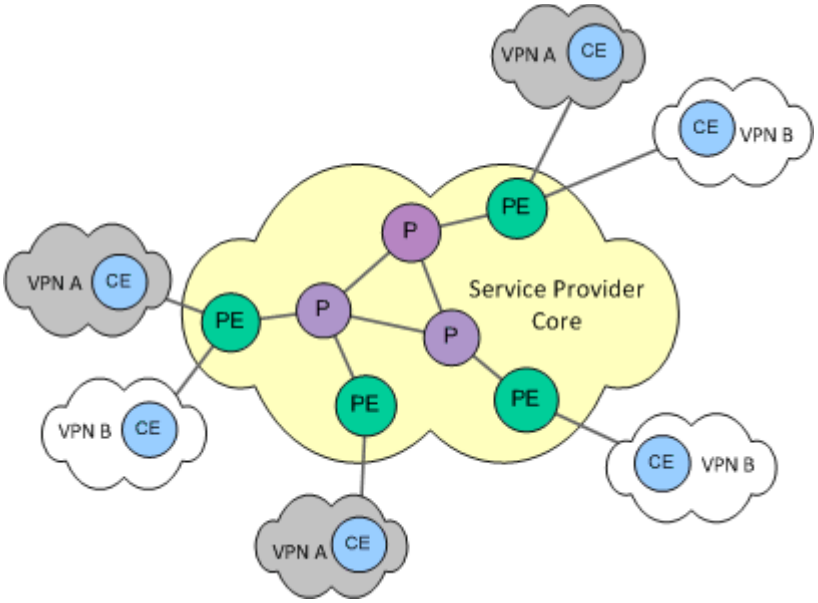
Roles of Routers

In an MPLS VPN, routers are classified according to the roles they perform.

- **Provider Edge** (PE) routers are those routers within a Service Provider core network that connect directly to a router at the customer's site.
- **Provider** (P) routers are all those routers within the VPN core network that are not edge routers.
- **Customer Edge** (CE) routers are those routers at the customer premises that have a direct interface to the PE router. These are sometimes known as CPE (Customer Premises Equipment) devices.

These roles are shown in Figure 4–1.

Figure 4–1 PE, P, and CE Routers



The Service Provider is responsible for managing the backbone, comprising the PE and the P routers. From an MPLS point of view, all routers in the core are LSRs. The PE devices are edge LSRs, responsible for attaching label headers at ingress and stripping labels at egress.

As far as MPLS VPNs are concerned, the CE devices are the responsibility of the customer, and they do not need to run MPLS. However, in cases where they are managed by the Service Provider, IP Service Activator can configure policy-based services on them.

BGP

The various routers within the VPN communicate using BGP, an IP routing protocol that defines how routes can be distributed. BGP transports information about CE routers only to members of the same VPN, ensuring security.

A BGP Autonomous System (AS) is a collection of networks under a common administration that share a common routing strategy. BGP communication that takes place within an autonomous system (for example, between PE routers within a Service Provider network) is known as Internal BGP (iBGP). BGP communication between different autonomous systems, for example, between a PE and a connected CE, is known as External BGP (eBGP).

Each AS is identified by an Autonomous System Number (ASN), which is required when running BGP. Service Provider networks normally use BGP, so they will already have an assigned ASN. However if eBGP is used as the routing protocol between PEs and CEs, a unique ASN must be assigned to each separate customer site network.

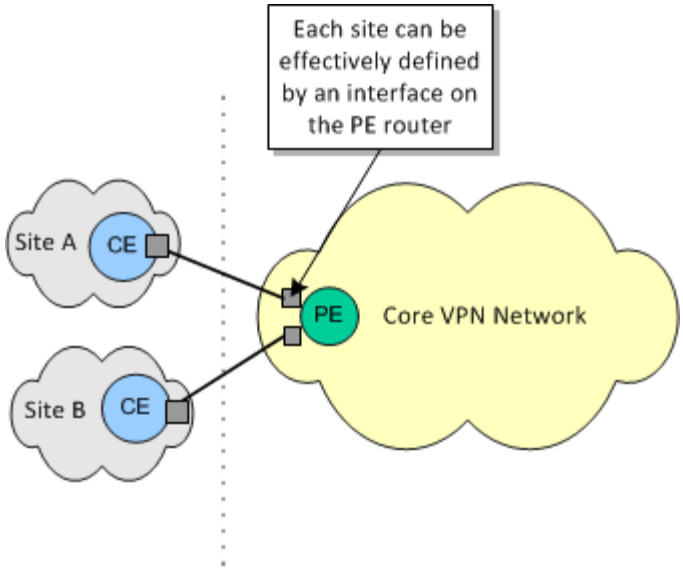
Site and Route Distinguishers

A VPN comprises a number of customer sites communicating over a shared network infrastructure.

A CE device is always in a single site, but a site may belong to multiple VPNs. A PE router can be connected to CE devices in a number of different sites, in the same or different VPNs.

Each customer site can be identified by its CE router, but from the Service Provider's point of view (who may not have visibility of customer equipment), each site is defined by interfaces on the PE router connected to the CE router, as in Figure 4–2.

Figure 4–2 Sites Defined by Interfaces



MPLS VPNs enable customer site networks to use overlapping address space, rather than globally registered IP addresses. This is achieved by ensuring unique addresses for use within the VPN, by means of Route Distinguishers (RDs), also known as Route Descriptors.

VPN-IPv4 and IPv6 Addresses

The PE adds the site RD to the IP address of each node within the customer site to create an extended address, known as a VPN-IPv4 address and VPN-IPv6 address. This is used to identify the node throughout the VPN. The VPN-IPv4 address and VPN-IPv6 address uniquely identifies each endpoint in the network, even if the customer site is using unregistered private IP addresses.

VRF Tables and Route Targets

Each PE router maintains a number of separate forwarding tables known as VRF (VPN Routing and Forwarding instance) tables, and each site (that is, each PE interface or sub-interface connected to a CE device) must be mapped to one of those VRF tables.

A VRF table does not necessarily correspond to a particular VPN. Its purpose is to hold the routes that are available to a particular site connected to a PE device. If a site is in multiple VPNs, the VRF table associated with that site contains routes for all the VPNs of which it is a member.

Routes defined within a PE's VRF tables are propagated to other PE devices within the same VPN. But since a VRF table is not mapped directly on to a VPN, it is necessary to identify the VPN to which each route applies.

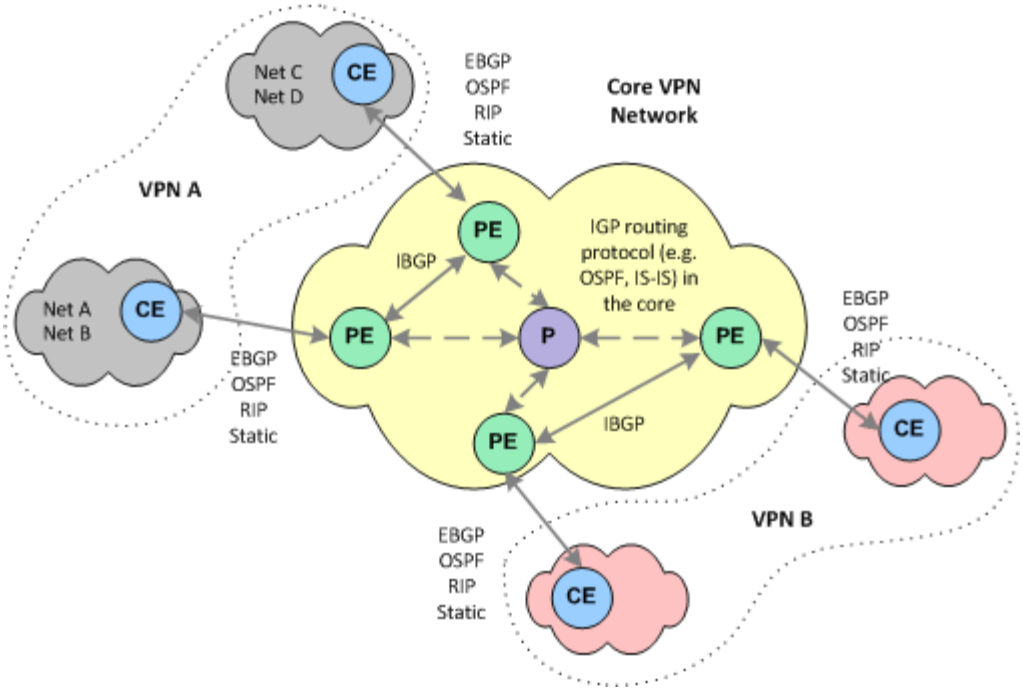
This is achieved by means of route targets. Every route that is distributed from a VRF table is tagged with an export route target attribute identifying its VPN. Each VRF is tagged with one or more import route target attributes, indicating the VPNs for which it wants to import routes.

When routes are distributed, any route marked with a particular export route target attribute will be installed in VRF tables marked with the same import route target attribute.

Routing Protocols in the VPN

The various routers within the VPN need to communicate using a routing information distribution protocol that defines who can talk to whom. Figure 4–3 illustrates the routing protocols required in an MPLS VPN.

Figure 4–3 MPLS VPN Routing Protocols



- PE routers that are connected to other PE routers within the same VPN communicate using iBGP. iBGP is used to transport VPN-IPv4 and VPN-IPv6 network addresses across the core network to other VRF tables.
- Each PE router also needs to communicate with its external neighbors (the CE routers to which it is connected) in order to transport these private routes between the CE routing table and the PE VRF table. IP Service Activator supports eBGP, OSPF and RIP routing protocols as well as static routing.

- In addition to BGP, an Interior Gateway Protocol (IGP), such as OSPF or IS-IS is required within the core network, to enable normal connectivity throughout the core, so that LSPs can be established.

VRF Tables

The IP Service Activator Network Processor and cartridges configure the VRF (VPN Routing/Forwarding Instance) tables and associated route targets on the PE devices. Each customer site connects to a PE interface or sub-interface. This interface is assigned to a VRF table, which defines the VPN membership of a customer site. VRF tables hold routing information that defines how packets from a given site are routed across one or more VPNs to other sites. They are private routing tables containing IPv4 and IPv6 routes that have been learned from CE routers using eBGP, RIP or OSPF and any explicitly defined static routes. They do not form part of the PE router's own routing tables.

VRF Table Names

VRF tables are generally given default names by IP Service Activator. However, you can define specific names for VRF tables on selected interfaces if you wish.

VRF Re-use and Reduction

A VRF table is set up on the device for each PE interface that is a member of a VPN. However, if multiple VRF tables contain exactly the same routes IP Service Activator will normally reduce them to just one, in order to minimize resource usage. This is known as VRF re-use or VRF reduction.

In some cases automatic VRF re-use may not be required. For example, you may want to provision dual links to customer sites in order to implement load balancing, requiring a separate VRF table for each connecting interface, or to reduce the impact of future re-configuration. In this case you can override VRF re-use by specifying that particular interfaces are always to have their own VRF table.

RD Number Formats

The RD number can be in either of the following formats:

- 32-bit IP address:16-bit number
- 16-bit ASN number:32-bit number

IP Service Activator normally generates RD numbers automatically, using the second of these formats. However you can override the default and specify your own RD numbers if you wish.

RT Number Formats

A route target (RT) identifies a set of sites within a VPN to which a PE device distributes routes. Route targets are used to create the VPN topology. Each VPN must have a unique route target number.

The RT number can be in either of the following formats:

- 32-bit IP address:16-bit number
- 16-bit ASN number:32-bit number

IP Service Activator normally generates RT numbers automatically, using the second of these formats. However, you can override the default and specify your own RT numbers.

VPN Topologies

The connectivity of the VPN can be one of the following:

- **Mesh**: All sites have connectivity to all other sites
- **Hub and Spoke**: One or more hub sites has access to all other sites; spoke sites can access the hub only
- **Management**: Works in the same way as hub and spoke, but is used when setting up QoS to ensure connectivity to CE devices

In a hub and spoke VPN topology, IP Service Activator generates two RT numbers – one for the hub site(s), generated as indicated above, and one for all spoke sites, generated by incrementing the hub low order number by 1.

You can specify your own RT numbers for hub, spoke or fully-meshed sites within a VPN if you do not want to use the default values. You can easily reassign RT numbers to sites within a VPN, if for example, it has been imported or exported by a different application.

You can specify any number of RT values per VPN and specify whether a value is used for VRF import, VRF export, or neither for hub, spoke, and fully-meshed behaviors.

RDs per VPN

By default, IP Service Activator automatically generates a site-specific VRF table name and RD number for each site that participates in a VPN.

However, you can override the IP Service Activator default by specifying at the VPN level that the same VRF table name and RD number is applied to all sites that participate in the VPN. You can choose whether to use IP Service Activator-generated values or specify your own VRF table name and/or RD number.

VRF Route Limit

The route-limit features enable you to specify the maximum number of routes that can be imported into the VRF table. Two alternative actions can be defined:

- A warning message can be generated if the number of imported routes reaches the maximum, but with routes continuing to be accepted.
- A warning message can be generated when the number of routes reaches a specified percentage of the maximum, with no further routes accepted when the maximum is reached.

Co-existence with Predefined VRF Tables

If an MPLS VPN has already been manually configured on a network, IP Service Activator is able to work with the pre-configured VRF tables that exist on devices. You can choose whether IP Service Activator ignores existing VRF tables or assumes control of them, and if existing content is preserved or removed.

Predefined Export Maps

You can apply a user-defined export map to the export policy configured by IP Service Activator. The export map exports only the VRF table routes whose prefixes match those specified in the export map to other PE devices. The export map tags these routes with only the RT numbers of sites that need to receive those routes.

PE-PE Routing -iBGP

iBGP is the protocol used for communication of VPN routes between PE devices in an MPLS VPN. In order for devices to exchange routing information, an iBGP session must be configured between the PE devices that comprise the VPN.

iBGP Peering Optional

If iBGP peering is already installed on PE routers, for example if Route Reflectors are used, the existing configuration can be preserved.

Load Balancing

You can enable load-balancing between iBGP peers by specifying the number of alternative routes to a given prefix that are maintained in a device's routing table. By default, this option is disabled and all identical routes learned from peer devices are dropped. To enable load-balancing, you can specify the number of routes that are maintained.

PE-PE Community Attributes

You can specify that routes advertised to the neighbor CE router contain the standard community attribute as well as the extended community attribute which is configured by default.

MD5 Authentication

The identity of iBGP peers and the integrity of data exchanged during iBGP sessions can be verified using MD5 authentication. This option uses the MD5 digital signature algorithm and a specified key of up to 255 characters to generate a checksum of the iBGP data that is to be sent from a PE device to its peer. The iBGP data and its checksum are then sent to the peer device using TCP. The recipient device uses MD5 and the same key to generate a checksum of the received iBGP data. If both checksums match, the identity of the sender and the integrity of the received iBGP data is verified.

PE-CE Configuration

In order to exchange information to and from customer sites in the VPN, each PE router also needs to communicate with each of its external neighbors – the CE routers to which it is connected.

The effect is to advertise network reachability information between the CE and the PE, which in turn converts IPv4 and IPv6 addresses to VPN-IPv4 and VPN-IPv6 addresses respectively, for traffic passing from the CE to the PE and from the PE to the CE.

PE-CE Protocols

IP Service Activator supports eBGP, RIP and OSPF. Static routes can also be configured in conjunction with eBGP, RIP and OSPF routing or used alone to define routing between the PE and CE.

Route Redistribution

IP Service Activator offers a high level of control over the redistribution of routes between protocols, enabling you to specify the metric to associate with routes distributed from the selected PE-CE routing protocol into other Internal Gateway Protocols (IGPs) and BGP, and vice versa.

Redistributing routes between protocols brings with it the risk of introducing routing loops and convergence problems. However, you can filter and refine the redistribution of routes by associating a manually pre-configured route map/policy statement with redistributed routes.

eBGP Configuration

You can make the following configurations to eBGP:

- **AS Override**: You can specify that the ASN of a provider is used to override the ASN of a site. In this case, a PE device that receives route prefixes whose AS_PATH attributes have one or more ASNs matching the ASN of its neighboring CE devices, substitutes those ASN instances with the ASN of the Service Provider network. Prefixes with the substituted ASNs are then re-advertised to neighboring CE devices. The PE device also adds its ASN to routes before exporting them to the CE device.

  This allows CE devices to accept routes that have been re-advertised by devices having the same ASN, and which would otherwise be rejected. Normally, a CE device rejects routes whose AS_PATH attribute contains ASNs matching its own ASN, to prevent routing loops.
- **Allow AS in**: You can specify the maximum number of times the same ASN is allowed to occur in the AS_PATH attribute of a route prefix advertised to the PE device for the prefix to be permitted and then re-advertised to neighboring CEs by the PE device.
- **Extended and Standard community attributes**: You can specify that routes advertised to the neighbor CE router contain the standard or extended community attribute or both.
- **Local preference**: Where multiple PE interfaces are associated with a site, the local preference for an interface can be set.
- **PE-CE authentication**: the identity of eBGP peers and the integrity of data exchanged during eBGP sessions can be verified using Authentication.
- **Prefix filters**: You can set up a prefix list file to specify that routes whose prefixes match those in the prefix list are either allowed or rejected by the PE router depending on their designation in the prefix list.
- **Prefix limit**: You can specify a maximum number of eBGP IP route prefixes that can be received by the PE router from its CE neighbor.
- **Site of Origin**: Site of Origin (SOO) is configured automatically for sites that have more than one CE to PE connection. It identifies the site from which the PE router learned the route and prevents routing loops from occurring when a site is multi-homed.
- **Multiple-path load sharing**: You can enable load-balancing between eBGP peers by specifying the number of alternative routes that are maintained in a device's routing table. By default, this option is disabled and all identical routes learned from peer devices are dropped.
- **Route dampening**: Route dampening is a mechanism that attempts to minimize network instability by suppressing the advertisement of poorly-behaved routes. Penalties are applied when a route is withdrawn, readvertised, or changed. When a predefined penalty limit is reached, further advertisement of the route is suppressed. The penalty is reduced according to a defined "half-life" setting, and once the penalty decreases below a limit, the route can be readvertised.

IPsec

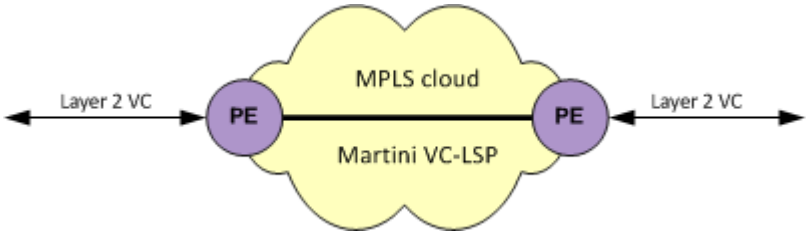
IP Service Activator allows you to configure VRF-Aware IPsec tunnels using configuration policies, which support Customer IPsec Access and Public IPsec configuration.

The VRF-Aware IPsec feature allows you to map IPsec tunnels that terminate on a shared public interface to specific Virtual Routing and Forwarding (VRF) instances. This allows you to map IPsec tunnels to MPLS VPNs extending customer VPN access to users that are not directly reachable via dedicated WAN links.

Martini Layer 2 MPLS VPNs

A Martini Layer 2 point-to-point connection is a Pseudo-Wire (PW) or tunnel configured between two endpoints across an IP network, as shown in Figure 4–4.

Figure 4–4 Layer 2 Martini Connection



The connection uses MPLS labels to encapsulate and transport various Layer 2 data formats, including VLAN to VLAN, Ethernet, Frame Relay, ATM Cell and ATM AAL5, across an IP network. The tunnel provides a transparent connection, so users see no change in their Layer 2 data. The tunnel does not aim to meet QoS aspects of the connection, particularly in the ATM case.

The Martini endpoints can be interfaces, sub-interfaces, or other endpoint identifiers (VCI/VPI on ATM, DLCI on Frame Relay, or VLAN ID on Ethernet.

A Martini Layer 2 VPN is an association of Martini Layer 2 point-to-point connections.

Implementation Scenarios

A Layer 2 Martini MPLS VPN enables the encapsulation and transport of legacy data types over IP networks. As Service Providers upgrade their network core, connections between legacy networks can be maintained. Customers needing traditional connectivity over a third-party network can be served using the same IP core network, regardless of the packet types they need to transport. Additionally, the tunnel saves the complexity of carrying the customers routes across the network.

Support of Ethernet technologies also permits Service Providers to use inexpensive Metro-Ethernet solutions in the Local Area Network (LAN), reducing the rollout cost of new networks.

Similarly, Martini solutions can reduce the rollout costs associated with mobile networks in transition from 2G to 3G. Using Martini tunnels, legacy 2G connection-oriented networks can traverse the new 3G IP core network. This saves the operational costs of supporting two different networks.

Layer 2 Martini VPNs on Routers and MPLS-enabled Switching Devices

IP Service Activator supports the configuration of Layer 2 Martini VPNs on MPLS-enabled switching devices, which encapsulate and transmit a number of different types of data.

These devices can be roughly categorized as either MPLS-enabled switching devices or routers.

MPLS-enabled switching devices support Layer 2 and Layer 3 switching features, MAC learning, and VLAN bridging. Table 4–1 lists the data types that can be encapsulated on switching devices.

Table 4–1 Layer 2 Martini VPNs on MPLS-enabled Switching Devices

Encapsulated Data	Endpoint	Comments
Ethernet (Port)	Ethernet port	Martini VLAN ID header is stripped on the pseudo-wire and re-applied (if required) on the exit interface.
Ethernet (VLAN)	VLAN endpoints configured under Ethernet interfaces (not sub-interfaces)	N/A

Routers support none of the switching features described above but support standard IP routing between interfaces. Table 4–2 lists the data types can be encapsulated on router equipment.

Table 4–2 Layer 2 Martini VPNs on Routers

Encapsulated Data	Endpoints	Comments
Ethernet (Port)	Ethernet interfaces	All VLAN tags are preserved across the connection. Frames that enter the tunnel labeled VLAN n leave the tunnel labeled **VLAN** *n*.
Ethernet (VLAN)	VC identifiers	The VC identifier value represents the VLAN ID. The same VLAN ID must be used at both ends of the connection.
ATM Cell	Sub-interface with VC identifier	N/A
ATM AAL5	Sub-interface with VC identifier	N/A
Frame Relay	Main interface with VC identifier	The VC identifier value attached to the main interface must be created manually.

Creating a Layer 2 Martini VPN in IP Service Activator

The high-level steps to create a Layer 2 Martini VPN in Service Activator are:

- Add the Layer 2 Martini connections.
- Set the options in the L2 Martini Pt-Pt property page.
- Assign the endpoints to the new Layer 2 Martini tunnel.

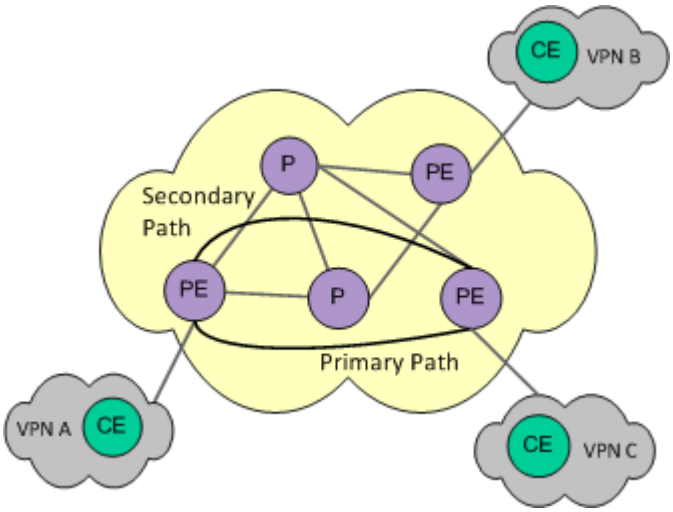
Complete details, including concepts, prerequisites, and procedures, are provided in *IP Service Activator VPN User's Guide*.

Layer 2 MPLS Label Switched Paths

Layer 2 MPLS Label Switched Path (LSP) Activation enables the provisioning of engineered LSP tunnels. An LSP tunnel is an entity that exists on a specific network device and identifies a hop-by-hop path through the network to another specific device. Once an engineered tunnel is in place between device A and device B, traffic flowing from A to B should travel through the network more efficiently.

A primary LSP on the Service Provider network is configured to be the preferred path. An optional secondary path provides path protection, as shown in Figure 4–5.

Figure 4–5 Optional Secondary LSP



In addition to the IP addresses of all the hops (between 0 and 30) in both the primary and backup (secondary) path between the 2 devices, the user can provision a number of attributes which affect the operation of the tunnel.

Supported Services

IP Service Activator supports the following Layer 2 MPLS Label Switched Path (LSP) Activation functionality on Cisco devices:

- LSP with primary and secondary paths
- Explicit paths
- LSP parameters:
  - bandwidth
  - hold and setup priorities
  - path option
  - affinity
  - IGP metric
  - fast-reroute

Metro Ethernet Virtual LAN (VLAN) Services

Ethernet has become the technology of choice for Metro networks, and is gradually taking over Frame Relay and ATM for access into the IP core network. IP Service Activator supports the ability to deploy and manage Metro Ethernet Virtual LAN (VLAN) Services in a multi-vendor environment, based on the 802.1Q standard. For example, IP Service Activator supports VLAN services on Cisco Catalyst IOS devices and Extreme Networks' Alpine, Summit and Black Diamond 6800 equipment.

Ethernet services are provisioned differently than IP services. Ethernet network services are configured at the edge and the core on a per customer basis. IP services are typically configured only at the edge of the Service Provider IP network. For this reason, Oracle Communications is delivering two levels of VLAN application, targeting the core network and edge access into the Ethernet network.

VLANs are configured on Metro Ethernet networks in IP Service Activator using a policy-based approach. A network object is created in IP Service Activator to represent the Metro Ethernet network. All devices in the Metro Ethernet network would be included under the network object. An appropriate custom role is applied to these devices and to the interfaces which are trunk port connections within the VLAN.

One configuration policy is used to define the VLAN IDs on the Metro Ethernet network and a second is used to apply trunk port configuration to the appropriate interfaces. The configuration policies are applied at the network level and role-based inheritance ensures that the configuration is distributed to the appropriate devices and interfaces, and applied as device configuration commands by IP Service Activator.

This powerful policy-based approach greatly reduces potential configuration errors and makes maintenance of the VLAN configuration easy.

Another advantage is that as additional devices are added to the Metro Ethernet network, you need to only discover them and apply appropriate device and interface roles, and all the required VLAN configuration is automatically applied. In case, you need to remove or add support for a single VLAN ID, the change is made in one place (on the vlanDefinitions configuration policy) and is automatically updated throughout your network.

VLAN Options

IP Service Activator supports multiple VLAN options:

- **Untagged**: Customer traffic comes into the Service Provider as plain, untagged Ethernet traffic. The edge switch and access port then tag the frames with the appropriate VLAN configured against that site. This allows the frames to be transported across the Ethernet core where the VLAN is enabled.
- **Tagged**: Customer traffic comes into the Service Provider with its own set of VLAN tags. In other words, the customer network is already segregated into VLANs which must be transported across the Service Provider network. IP Service Activator supports the configuration on that site of the specific set of VLANs to match the specified incoming traffic.
- **Stacked VLANs** (or **Q-in-Q**): A Service Provider can carry up to 4096 VLANs within a single Metro Ethernet network. If each customer were to connect with tagged traffic for multiple VLANs, the Service Provider could run out of VLAN IDs for all of its customers. The Stacked VLAN solution can prevent this from happening. Using Stacked VLANs, the Service Provider can encapsulate incoming customer-tagged traffic into a single VLAN, and remove the encapsulation at the destination site. This approach can transport a whole set of customer VLANs

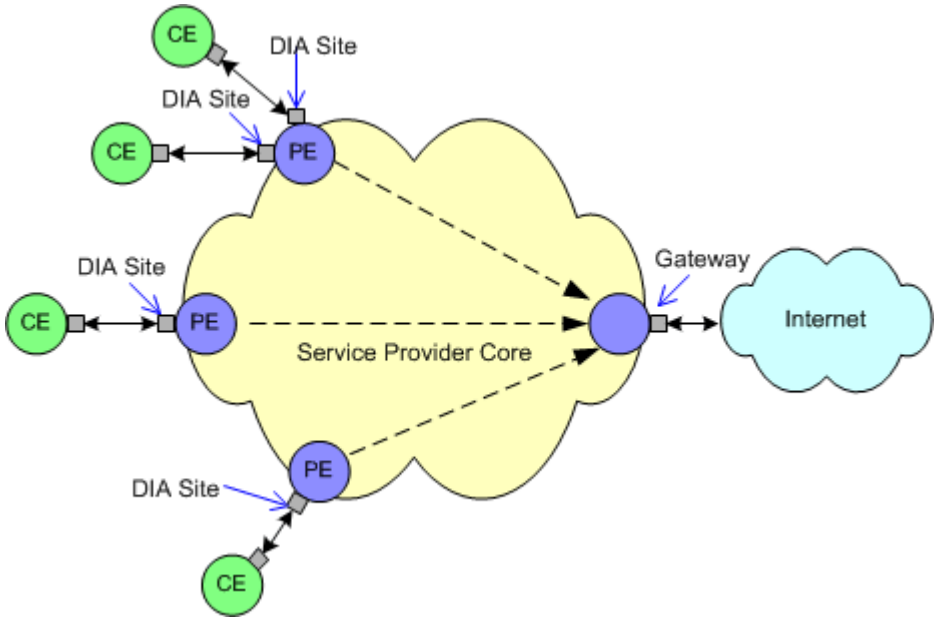
through a single Service Provider VLAN. Stacked VLANs can support over 16 million VLANs in a single Metro Ethernet network.

Dedicated Internet Access

IP Service Activator supports Dedicated Internet Access (DIA) Services through a combination of capabilities built into the IP Service Activator service model and the application of interface and routing configuration policies. This is done through the configuration of Layer 3 interfaces on the PE and the configuration of basic routing (typically static routes) to provide reachability to Internet gateways.

Figure 4–6 illustrates a typical scenario in which it is required that each customer site be able to access the Internet. This is accomplished through the use of DIA sites on the connected PEs.

Figure 4–6 DIA Sites



A Layer 3 interface is configured or created and placed in a DIA site. In a managed CE service, the CE device is also placed in the DIA site. IP routing is configured on the PE to route Internet traffic across the provider core to the Internet gateway and on to the Internet. Additionally, prefix lists may be configured to filter routing updates.

Each DIA site represents a physical location and includes an access interface on a PE device, and a CE device. It is used to link the CE device to the PE interface so that routing to the Internet gateway can be provided.

For each of these DIA sites, you need to create a logical Layer 3 interface on the PE dedicated to that site. Each Layer 3 interface will have a different IP address (because they are public IP addresses), and will connect to the service provider public network routing.

Layer 3 Multicast Services

Multicast provides an efficient means of transmitting the same data to multiple receivers in a multicast group. A multicast group is an arbitrary number of receivers that join a group in order to receive a particular traffic. This multicast group has no

physical or geographical boundaries—the hosts (sources or receivers) can be located anywhere on the Internet or on any private internetwork. A group of receivers can listen to a single address, or a group of addresses.

A multicast address identifies multiple interfaces, and is used for one-to-many communication. With the accurate multicast routing topology, packets addressed to a multicast address are delivered to all interfaces that are identified by the address. A packet sent to a multicast address is delivered to all interfaces identified by the multicast address, whether it is IPv4 or IPv6.

Multicast uses the class D address range running from 224.0.0.0 through 239.255.255.255 for IPv4 multicast addresses. The addresses do not have a subnet mask length associated with them for forwarding purposes. Each address is treated independently so the subnet mask used for forwarding is always assumed to be /32.

IPv6 multicast addresses have the Format Prefix of 1111 1111. An IPv6 address is simple to classify as multicast because it always begins with FF. Multicast addresses cannot be used as source addresses.

You can use a subnet mask only for discovery using IPv4, not for discovery using IPv6.

Applications that take advantage of multicast include video conferencing, corporate communications, distance learning, and software distribution.

IP Service Activator supports two Layer 3 multicast services:

- IP multicast
- VPN multicast

IP Service Activator configures multicast using configuration policies.

IP Multicast Services

IP Service Activator supports configuration policy-based activation of IP multicast on Customer Edge (CE) routers. This includes the configuration of multicast RPs using static configuration, Auto-RP, or Bootstrap Router (BSR). Configuration of both Protocol Independent Multicast Sparse Mode (PIM-SM) and Protocol Independent Multicast Source Specific Multicast (PIM-SSM) are supported as well as activation of IGMP, Cisco Group Management Protocol (CGMP), and Router-Port Group Management Protocol (RGMP) on LAN interfaces.

IP multicast is configured on CE interfaces and is supported on specific Cisco IOS devices. Support can be extended to other devices using IP Service Activator Software Development Kit (SDK).

VPN Multicast Services

VPN multicast provides the ability to support multicast over a Layer 3 Virtual Private Network (VPN). As Layer 3 VPNs support only unicast traffic connectivity, deploying this service in conjunction with a Layer 3 VPN allows service providers to offer both unicast and multicast connectivity to Layer 3 VPN customers.

VPN multicast allows an enterprise to transparently interconnect its private network across the network backbone of a service provider. The use of a VPN multicast to interconnect an enterprise network in this way does not change the way the enterprise network is administered, nor does it change general enterprise connectivity.

IP Service Activator supports:

- A set of policies that enable the creation of VPN multicast sites.

- Creating Access Lists when configuring VPN multicast. The access lists include:
  - PE-CE Access List and PIM
  - Mrinfo-filter
  - Static RP-to-Group mapping
  - PIM-SSM Range
  - SPT Threshold
- Enabling VPN multicast on a device
- Setting Rate-Limit on PIM-SM register messages
- Configuring the address of a PIM rendezvous point for a particular group
- Defining PIM-SSM range
- Specifying the threshold that must be reached before moving to the SPT

VPN multicast is configured on Provider Edge (PE) interfaces and is supported on compatible Cisco IOS devices. The support can be extended to other devices using IP Service Activator SDK.

5

Policy-based Services

This chapter explains the policy-based service activation components that can be used to create a service to users.

Oracle Communications IP Service Activator supports a variety of policy-based services for a range of devices from different vendors. For more information about the services supported for each device, refer to the supported features tables in the following cartridge guides:

- *IP Service Activator Cisco CatOS Cartridge Guide*
- *IP Service Activator Cisco IOS Cartridge Guide*
- *IP Service Activator Cisco IOS XR Cartridge Guide*
- *IP Service Activator Huawei Cartridge Guide*
- *IP Service Activator Juniper JUNOS Cartridge Guide*

Key Policy Configuration Concepts

IP Service Activator uses a policy-based approach to implement QoS, access control and selected measurement features. It is a flexible and powerful model, based on two key concepts:

- **Policy elements**: The definitions of the policies to be applied. Policy elements define what policies are configured.
- **Policy targets**: The points in the network at which the defined policies are applied. Policy targets define where policies are configured.

These two aspects of setting up a policy are described in the following sections.

About Policy Elements

In IP Service Activator, you can set up specific policy elements, which can then be applied to the relevant points in the network. A policy element can be one of the following:

- Policy Rules are used to classify traffic and define packet marking, policing requirements, and access control.
- Per Hop Behavior Groups (PHB groups) are used to provide low-level control of the queuing, policing, marking, congestion avoidance and traffic shaping mechanisms available on a particular interface.

- SLA Measurement and Collector Parameters are used to define NetFlow or MIB-based measurement and to specify that NetFlow or SNMP MIB data is to be collected.

A configured policy can comprise a combination of these policy elements.

Policy Rules

One aim of policy-based network management is that high level policies can be defined by an organization's business requirements and the actual details of implementation – the conversion from a high-level request to the actual configuration of a network device – are hidden from users.

Although policy-based network management is commonly used to implement quality of service it can also be applied to resource allocation, security issues, measurement and other configuration requirements.

Policy rules can define one or more conditions and the actions associated with them, examples of which are shown in Table 5–1.

Table 5–1 Examples of Conditions and Actions for Policy Rules

Condition	Action
Trading traffic is travelling between New York and Chicago.	Specify a bandwidth of 128k
Web-browsing traffic transiting the core.	Mark as low-priority and drop in times of congestion
Game-playing traffic detected.	Deny access to identified traffic between 9:00 and 5:00

Within IP Service Activator there are three types of rule:

- **Classification rules** are used to identify traffic and mark packets. They also apply bandwidth limits to defined traffic.
- **Policing rules** are used to identify traffic and set up policing parameters – the bandwidth requirements and the action to take for conforming and non-conforming traffic.
- **Access rules** are a security measure that can deny or explicitly permit identified traffic to access the network.

All rules identify and classify traffic in the same way. In addition all rules can be made dependent on date and time.

Per Hop Behavior Groups

A PHB group is a way of applying one or more policy definitions to a particular interface. There are two types of PHB group:

- Standard PHB groups are used to implement QoS mechanisms (queuing, congestion avoidance, policing and traffic shaping) on Cisco and Juniper devices. Although where possible these are generic, such as Weighted round Robin (WRR) the actual forwarding behavior is specific to particular device manufacturers and types.
- MQC PHB groups allow you to implement Modular QoS CLI mechanisms developed by Cisco to simplify the configuration of QoS on all device types.

SLA Measurement and Collector Parameters

Measurement parameters and collector parameters are used when configuring SLA monitoring. For more information, see "SLA Monitoring".

About Policy Targets

A policy target is the point in the network at which a policy element is to be applied. Depending on the policy, this can be a device, an interface, a sub-interface or a PVC endpoint.

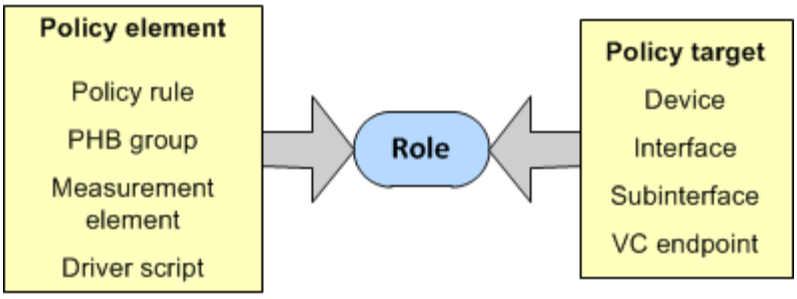
IP Service Activator's policy model is completely flexible: policy elements can be defined at any point in the system, but the policy targets at which they will be implemented depend on the use of policy roles and an inheritance model.

Policy Roles

Roles enable you to logically group devices and interfaces, for example by customer or service package, and set up policy specifically targeted at that group. A role is a way of grouping a set of policy targets so that they attract the same policy-based configuration.

As illustrated in Figure 5–1, a role identifies the function of a configurable network object such as a device or an interface. These roles can then be linked to policy elements (such as rules, PHB groups or driver scripts) to define the policy configuration that is to be applied. Policy elements are only applied to network objects that have matching roles.

Figure 5–1 Assigning Roles to Policy Targets and Elements



Each device and interface to be managed using IP Service Activator must have at least one allocated role which defines its function.

IP Service Activator includes a number of pre-defined device and interface roles that define how it works. Pre-defined roles are important because they ensure that VPN configuration is applied to the appropriate devices and interfaces.

The pre-defined roles support a DiffServ model, consisting of Access, Gateway, and Core devices:

- Access devices are routers on customer premises that provide access to the core Service Provider network. Access devices are equivalent to Customer Edge (CE) routers in MPLS terminology.
- Gateway devices are those on the edge of the core network that have a direct connection to the local or customer access device. Gateway devices are equivalent to Provider Edge (PE) routers.
- Core devices are those used for routing within the core Service Provider network. Core devices are equivalent to Provider (P) routers.

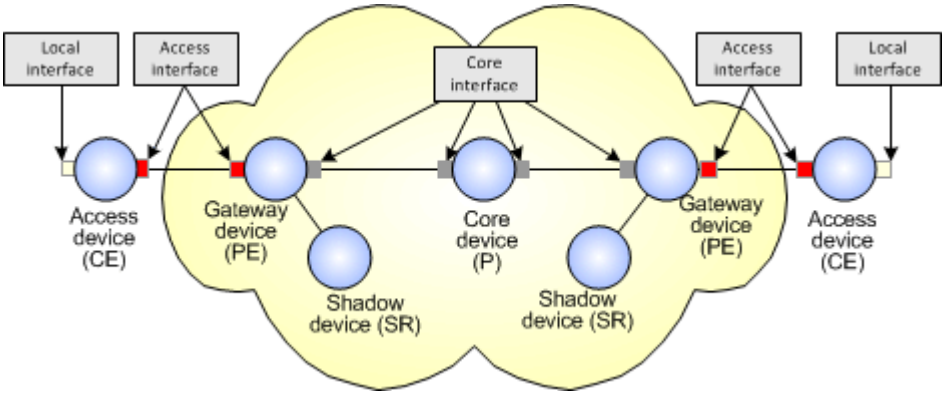
In addition, a Shadow device role is pre-defined; this is used when setting up Service Assurance Agent (SAA) measurements.

There are also four system-level interface roles:

- Access interfaces connect access and gateway routers
- Local interfaces are interfaces on access routers that connect to the customer's local LAN or hosts
- Core interfaces connect core and gateway routers or two core routers
- Disabled interfaces are not used.

These system-defined device and interface roles are illustrated in Figure 5–2.

Figure 5–2 System-defined Device and Interface Roles



The system-defined roles allow you to set up a simple QoS policy based on DiffServ, but for a more complex policy, you can create any number of user-defined roles.

Types of interfaces or devices can be grouped together and given a specific role which can then be used to define the policy that is applied. The type of roles required depend on the policy to be applied. For example, roles could be based on interface capacity (64k, 128k, 1Gig), device function, customer or service package.

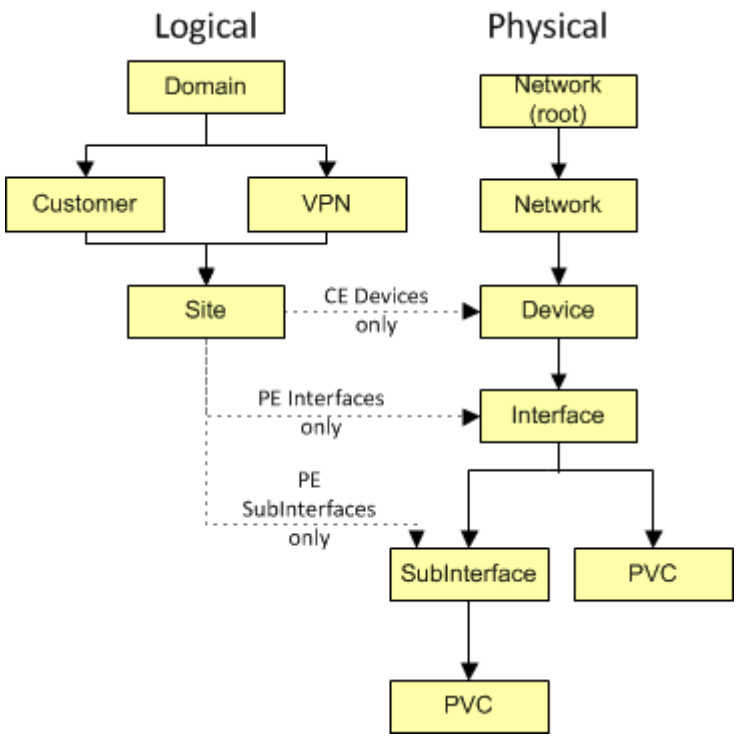
Inheritance

A process of inheritance means that a policy element that is to be implemented only needs to be applied at a single point in the network and will then automatically apply to all appropriate points. For example, a policy rule to mark packets can be applied to a network object and it will be implemented at all relevant interfaces with appropriate matching device and interface roles.

There are two branches in the inheritance model, illustrated in Figure 5–3:

- **Logical**: Includes domains, customers, sites and VPNs
- **Physical**: Includes networks, devices and interfaces

Figure 5–3 Inheritance Model



Configuring QoS Policies

Quality of Service (QoS) can be defined as a set of specific measures, characteristics and properties that defines how well a network is performing, as experienced by particular traffic flows across the network. QoS can be measured in a number of ways:

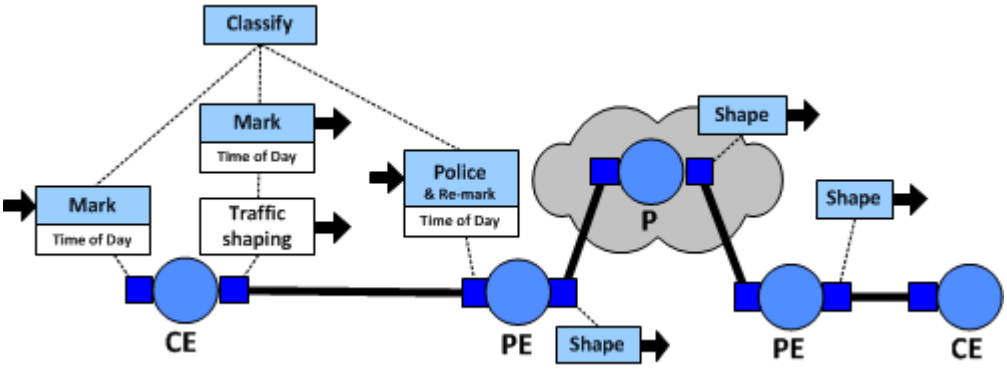
- **Delay or latency**: How long it takes for a packet to get from source to destination.
- **Jitter**: The variation in latency between subsequent packets. This is particularly important for audio or video transmissions.
- **Throughput**: The average and peak transmission rates.
- **Data loss**: How often packets are dropped and have to be re-sent.

Class of Service (CoS) techniques implement QoS by categorizing network traffic into a small number of defined aggregate classes. This enables some identified network traffic to be treated better than the rest; for example by allocating it more bandwidth, ensuring faster handling, or guaranteeing a lower than average loss rate.

Implementing a QoS solution requires a number of techniques, as illustrated in Figure 5–4:

- Identifying and classifying traffic to determine the QoS to be applied
- Marking the packets so that they can be recognized by nodes throughout the network
- Traffic shaping, to constrain specific outbound traffic to a particular bandwidth range
- Queuing techniques, to prioritize different traffic separately on outbound queues
- Policing traffic to ensure that traffic classes do not exceed their share of resources

Figure 5–4 QoS Implementation



IP Service Activator is also capable of setting the Diffserv Codepoints, IP Precedence or MPLS experimental bits where they are supported. Marking is implemented by means of classification rules.

Each classification rule defines a set of conditions and the actions that are taken if the conditions are true. The conditions can be any combination of the following:

- One or more classification types which identify the source, destination and type of traffic.
- Date and time (optional): specifies the period of time that the rule applies.

Where these conditions are true for a particular packet, a set of actions is performed.

Classifying Traffic

Routers need to identify and classify types of traffic that are to be allocated different QoS. For example, traffic can be identified by source or destination address or by source or destination port. For IP traffic, this requires the IP header to be examined. Ideally, traffic would be classified once only at the edge of the network, as some overhead is involved in checking the packet. Methods of classification depend on the capabilities of the router.IP Service Activator can classify traffic on any combination of:

- Source address
- Destination address
- Type of traffic

The identified source and destination can be a specific IP address or a subnet.

The traffic type is generally identified by source or destination port number and IP protocol, since all network devices can classify traffic in this way.

For devices that can support other methods of classification, IP Service Activator also supports traffic identification and classification by the following:

- Packet marking (such as IP Precedence settings or DiffServ codepoints)
- DNS domain name
- Application or sub-application for HTTP traffic, URL, MIME type, VLAN, or port

Traffic classifications are set up by means of traffic types, which can then be associated with the policy rule. A number of predefined traffic types identifying the most common TCP and UDP port numbers are included and additional ones can be set up as required.

Global classification categories can be set up, which identify a particular type of traffic, between a source and destination. Any number of these classification types can then be associated with a particular policy rule. Alternatively, the source, destination and traffic type can be defined specifically for each rule.

Marking Traffic

Identified traffic can be marked with a value to indicate the QoS/CoS that is to be applied to it. If CoS techniques are used, a small number of classes are defined, such as Gold, Silver and Bronze, with each one marked with a specific codepoint. Many marking types are supported, including: MPLS topmost, Frame Relay Discard Eligible, ATM cell loss priority, discard-class, trust types, COS, COS inner, and QoS group.

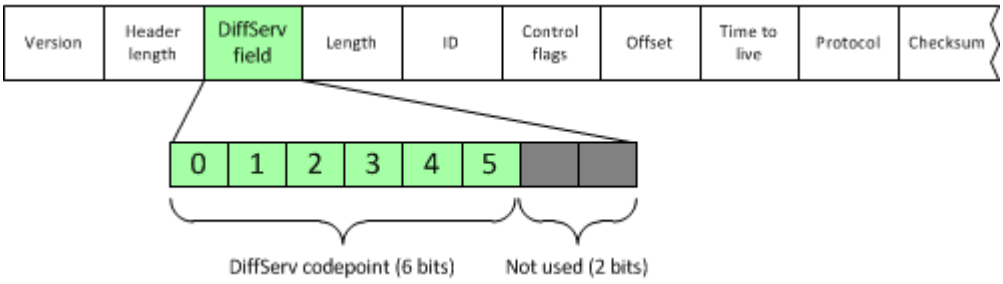
IP Service Activator supports the following standards for marking IP packets:

- DiffServ Codepoint
- IP Precedence
- MPLS Experimental Bits

DiffServ Codepoint

The RFC 2474 standard, "Definition of Differentiated Services Field in the IPv4 and IPv6 Headers", specifies that IPv4 packet headers include a DiffServ field, as illustrated in Figure 5–5.

Figure 5–5 DiffServe Field Header

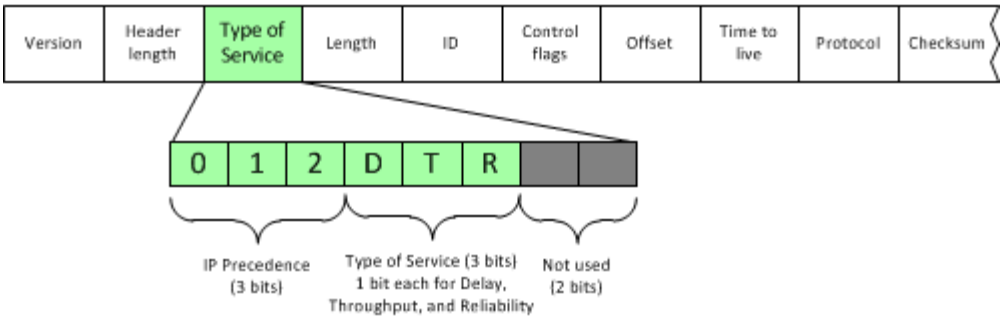


The six DiffServ bits allow the definition of up to 64 different classes of service. However, the standards define a number of recognized codepoint settings and the corresponding packet forwarding treatment, known as a per-hop behavior.

IP Precedence

RFC 791, which defines the format of IPv4 packet headers, specifies that IP packet headers should include a Type of Service byte, structured as illustrated in Figure 5–6.

Figure 5–6 Type of Service Header



The three-bit IP Precedence field is intended to indicate the importance or priority of the packet. It allows a set of eight values from 0 to 7 to be set, where 0 is the lowest priority and 7 is the highest. Values 6 and 7 are normally reserved for network control traffic, such as routing updates, leaving six values available for normal traffic.

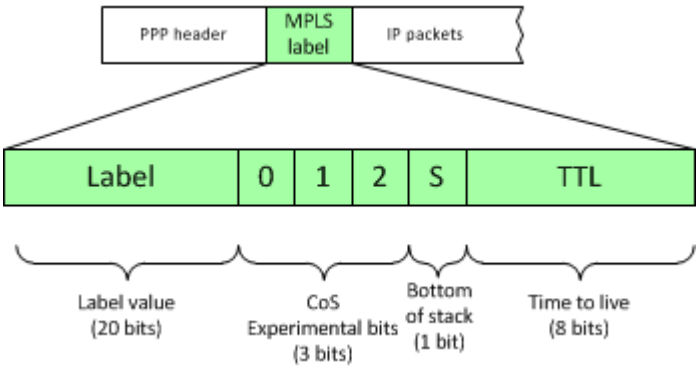
The three bits of the Type of Service field are intended to be used as switches which can be used in combination to specify requirements for Delay, Throughput and Reliability respectively.

MPLS Experimental Bits

For MPLS traffic, a label is attached to each IP packet at the ingress router to the LSP. This label is used by routers when forwarding packets along the path, and the IP packet header is not examined at any point along the LSP. The three IP Precedence bits are copied from the IP header into the three bits within the MPLS label known as the experimental CoS bits.

The application that generates the IPv4 packet controls the original IP Precedence value. However, some devices are able to reset this value, which can be useful if a precedence is set for a packet at the edge of the network and a Service Provider wants to override this value while the packet transits the core.

Figure 5–7 MPLS Experimental Bits Label



Shaping and Queuing Traffic

Traffic shaping and queuing techniques are implemented by means of PHB groups. A PHB group is a way of applying one or more policy definitions to a particular interface. The actual forwarding behavior is controlled by the QoS mechanisms that control the queuing and dropping of packets on each router, and thus are specific to particular device manufacturers and types. However, IP Service Activator's detailed reporting of device and interface capabilities ensure that it is easy to find out what

mechanisms are supported.The PHB group maps one or more defined classes of service on to the queuing and shaping mechanisms available at that interface.IP Service Activator supports the following queuing and traffic shaping mechanisms:

- Priority Queuing
- Weighted Round Robin (WRR)
- Weighted Fair Queuing (WFQ)
- Weighted Random Early Detection (WRED)
- Rate Limiting
- Frame Relay Traffic Shaping
- ATM Traffic Shaping

Traffic Shaping

Traffic shaping, or rate limiting, is a mechanism for controlling traffic on an interface by constraining specific outbound traffic to a particular bandwidth range, for example by specifying a particular maximum bandwidth or by defining burst parameters that the traffic cannot exceed.

Traffic shaping is often used to control access to the core network by constraining the outbound traffic. Traffic shaping does not in itself implement queuing, and in some cases can be used in conjunction with a queueing mechanism.

Generic Traffic Shaping (GTS) on Cisco routers is an example of rate limiting. Rate limiting is implemented in the form of a “token bucket” mechanism. The average bit rate defines the rate at which tokens are placed into a bucket of fixed capacity. Each token permits a number of bits to be transmitted. To send a packet, an appropriate number of tokens must be removed from the bucket. If there are not enough tokens in the bucket to send a packet, the packet is queued (or dropped if the queue is full). Therefore, the largest burst that can be transmitted is proportional to the size of the bucket.

Queuing Mechanisms

Throughout the network, QoS can be applied by the implementation of specific queuing mechanisms implemented at the various routers throughout the network.

These can be defined as congestion management techniques (applying a queuing algorithm to organize and prioritize the traffic), and congestion avoidance techniques (selectively dropping lower priority traffic to ensure that TCP slows down the rate of its transmission, in order to avoid congestion before it starts).

The way in which QoS mechanisms are implemented is specific to particular device types and models, and more detailed information can be obtained from the appropriate manufacturer. In some cases default settings can apply, and in others various parameters can be set.

Policing Traffic

Policing involves checking the traffic to ensure that the packets meet an agreed profile, for example an average and burst bit rate. Packets that are out of profile may be dropped or remarked to a lower class of service.

Policing is implemented by means of policing rules, which can be used to limit and optionally re-mark identified traffic. A rule defines the bandwidth and burst

requirements for the identified traffic, and the action to be taken if traffic conforms to or exceeds the requirements.

Each policing rule defines a set of conditions and the actions that are taken if the conditions are true. The conditions can be any combination of the following:

- One or more classification types which identify the source, destination and type of traffic
- Date and time (optional): specifies the period of time that the rule applies

The following bandwidth requirements can be set:

- Average rate permitted
- Maximum burst rate permitted
- Burst interval: the period of time over which the traffic is allowed to maintain its maximum burst rate

Policing actions can be defined for traffic that conforms to or exceeds its bandwidth allocation. Actions are:

- **Transmit**: Transmit packets as they are. This would be the normal course of action for conforming traffic.
- **Drop**: Drop packets immediately. You would normally only choose to drop packets where low priority traffic had exceeded its limitations.
- **Re-mark and Transmit**: Re-mark packets with a different packet marking and transmit. The packets may then be treated differently at subsequent routers.
- **Re-mark and Continue**: Re-mark traffic with a different codepoint and continue checking traffic against subsequent policing rules. This could be used when traffic exceeds bandwidth or burst rates. For example, if traffic marked as “Gold” exceeds the agreed policing, it can be remarked to “Silver”, and checked against subsequent rules; if this traffic then exceeds the “Silver” policing it can be remarked to “Bronze”

Modular QoS CLI

Modular QoS CLI (MQC) is Cisco’s simplified configuration of policy mechanisms and actions for traffic queuing, shaping, policing, congestion avoidance and re-marking. It is supported on selected devices and IOS versions.

IP Service Activator supports MQC by means of a specific MQC PHB group, which can be used to define an entire QoS policy that may be used at various points in the network. For example, an MQC PHB group might be used to manage the traffic going into the core network or to maintain the prioritization set up at the network edge throughout the core network.

An MQC PHB group may be applied to one or more classes of service that are based on a classification or classification group. The classification may be defined by factors such as source and/or destination IP address or account and traffic type. A number of classes of service may be linked to an MQC PHB group and one or more different mechanisms applied to each one.

The following traffic management mechanisms can be implemented within an MQC PHB group and applied to traffic by class of service:

- **Low Latency Queuing (LLQ)**: Assigns a traffic class to a strict priority queue with a guaranteed maximum bandwidth during congestion

- **Class-based Weighted Fair Queuing (CB-WFQ)**: Assigns a traffic class to a queue with a priority based on a guaranteed minimum bandwidth during congestion
- **Class-based Policing (single-rate or two-rate)**: Specifies and enforces conditions that define the maximum inbound and outbound bandwidth of a traffic class, packets that exceed the conditions can be dropped or re-marked
- **Class-based Shaping**: Specifies and constrains the maximum outbound bandwidth of a traffic class, outbound packets that exceed the conditions are delayed
- **Class-based Marking**: Marks packets to allocate a priority status or class
- **Congestion avoidance (Queue limit and/or WRED)**: Defines how packets are discarded during congestion

You can specify the order in which packets are evaluated against each CoS's match criteria to ensure that packets have the correct mechanism applied to them.

You can also nest an MQC PHB group within another MQC PHB group. This enables you to apply a policy to a broad range of traffic (defined by the parent MQC PHB group) and another to a subset of that range (defined by the child MQC PHB group).

Configuring Access Control Policies

IP Service Activator's access control features enable Service Providers to block specific applications, protect sensitive data or prevent identified users from accessing certain services. As part of a comprehensive security policy, this can protect sites and users from malicious denial of service attacks, prevent access to services by unauthorized users and prevent data loss.

Like QoS services, access control is implemented by means of rule-based policies that can deny or explicitly permit identified traffic. Rules can be set up to apply at specific dates or times

Access rules (or filters) are used to provide network security. Identified traffic can be denied or permitted access to the network. Each access rule defines a set of conditions and the actions that are taken if the conditions are true.

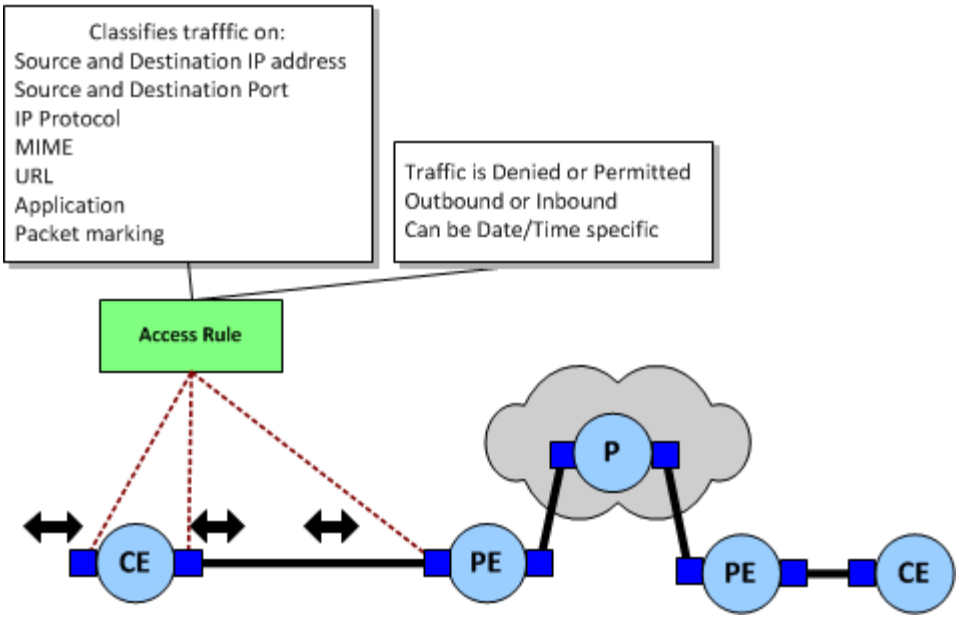
The conditions can be any of the following:

- One or more classification types which identify the source, destination and type of traffic
- Date and time: optionally, specifies the period of time that the rule applies

Where these conditions are true, the identified traffic can be denied or permitted access (either inbound or outbound or both).

Figure 5–8 illustrates how access rules work.

Figure 5–8 Access Rules



6

Extending IP Service Activator

This chapter explains features that can be used to extend Oracle Communications IP Service Activator's activation capabilities using the Configuration Template Module.

About the Configuration Template Module

The IP Service Activator Configuration Template module (CTM) lets you do the following:

- Create, update, and delete configuration templates.
- Activate a template.
- Perform related activities (view template events, set the lifetime of events, manage user access to CTM, and match templates to schema versions).

> **Note:** You can use Design Studio to import CTM templates, provided the templates are intended for use only in OSM.

If you want to use the CTM template from the IP Service Activator client, you can load the CTM template into the CTM server. If you want to use the CTM template as part of flow-through, use Design Studio to generate a service action for the CTM template and then make use of the service action in the activation task. For information about using CTM in Design Studio, see the Design Studio online Help.

CTM helps you 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.

CTM consists of three graphical user interfaces (GUIs):

- The **Activation GUI** is used to select a template appropriate to an object or set of objects, to fill in data forms, and to send the resulting configuration command set to objects in the network.
- The **Administration GUI** is used to show all templates, and to create, modify, view, and delete templates. Templates define the layout of the data entry forms and the configuration commands to be sent.
- The **Audit History GUI** is used to review administration events (create, update), provisioning events (activate) and event details.

These applications are launched from an object context within the IP Service Activator client. The launch object can be a network, device, interface, or sub-interface.

To run CTM, you must purchase and run the OSS Integration Manager (OIM). You must also run the IP Service Activator client, CTM Request Server, and Oracle database (SQL and Oracle listener).

7

Integration Features

This chapter describes the features of Oracle Communications IP Service Activator that enable it to integrate with other OSS applications.

REST Web Service API

The IP Service Activator installation provides an optional web service API that allows you to use the Representational State Transfer (REST) protocol in IP Service Activator to integrate with Oracle Communications Order and Service Management (OSM). The IP Service Activator REST web service allows you to create custom Groovy scripts that use common REST methods.

For more information about using the REST web service, see *IP Service Activator API Developer's Guide*.

Web Service API

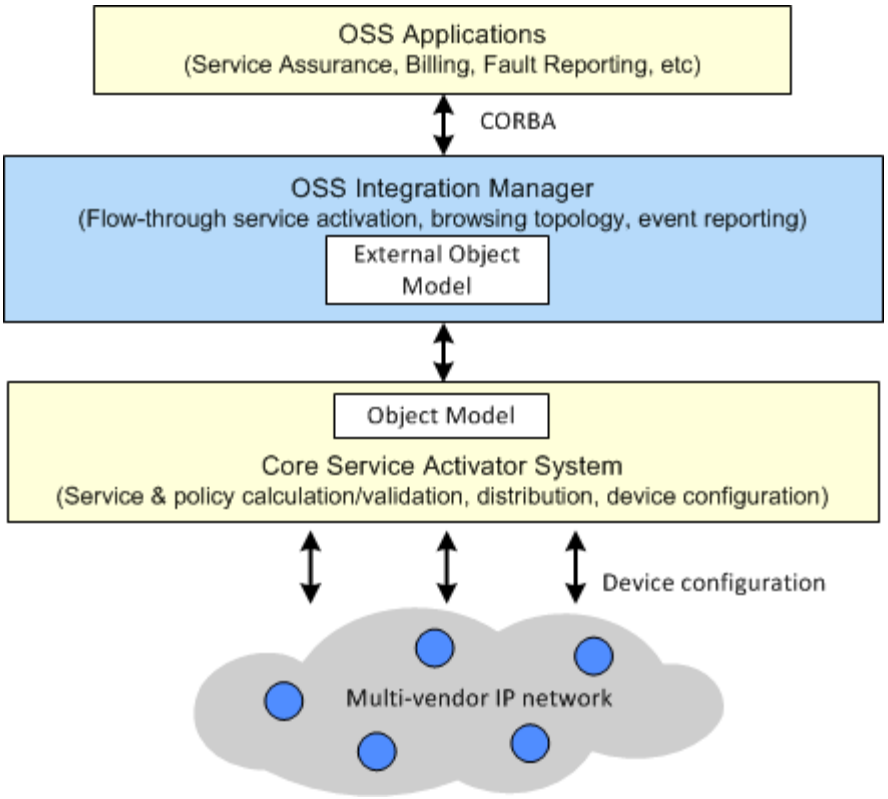
The IP Service Activator installation provides an optional web service API that allows IP Service Activator to integrate with Oracle Communications Order and Service Management (OSM). The IP Service Activator web service supports the use of multiple OIM instances connected to a single instance of IP Service Activator.

For more information about using the web service, see *IP Service Activator API Developer's Guide*.

OSS Integration Manager

The OSS Integration Manager (OIM) enables IP Service Activator to be easily integrated with OSS applications such as order entry, service assurance, fault management, and billing, as illustrated in Figure 7–1.

Figure 7–1 OSS Application Integration



The OIM consists of an open application program interface (API) that allows systems developers to access subsets of the IP Service Activator data model. Use of a Common Object Request Broker Architecture (CORBA) interface ensures secure and standardized communication with other applications, and can easily be adapted to use alternative architectures such as Tibco, Vitria, or XMP. Developers can develop scripts using any CORBA-compliant programming language (such as C++, Java, or Python), or use a command-line interface which allows user to run the commands interactively.

The OIM has been specifically designed to simplify the work of system developers wishing to integrate with IP Service Activator. To write applications to integrate with IP Service Activator, developers need:

- Details of the OIM command set and syntax
- Details of the objects that can be accessed and the relationships between them

About the OIM Command Set

The command language comprises a set of commands that provide an interface to the API. It is limited to a small set of generic commands such as **create**, **find** or **modify**, that operate on defined objects and their attributes. The use of generic commands means that the command set is not affected by changes to the object model, and is sufficiently flexible to support diverse uses such as integration of OSS applications, service activation or custom web development. Commands are grouped into various modules, but all commands have the same basic format, similar to UNIX or other scripting languages:

```
command [object_path] [parameters]
```

Commands that execute changes to the database and network are aggregated into transactions, which work in the same way as using transactions from the IP Service Activator client. This logical approach helps ensure straightforward scripting.

About the External Object Model

The External Object Model (EOM) is a simplified version of IP Service Activator's knowledge store. It defines all the objects that can be accessed or updated by external applications, including their attributes and the relationships between them.

Using a subset of the entire object model means that user programs can create and access data objects without needing to know the underlying complexity of the entire object model. The External Object Model is independent of the command syntax – new attributes and objects can be added without requiring new commands.

Objects in the External Object Model are divided into three major categories:

- **Topology**: These objects represent the topology of the managed network, such as Network, Device and Interface objects.
- **Policy**: These objects represent elements used to define policies and services that can be configured on network nodes, such as Domain, Rule and Traffic Type objects.
- **System**: These objects represent IP Service Activator components and the event handler subscription model, such as Component, Subscription and Fault objects.

Transaction Monitoring Functions

In IP Service Activator, the Policy Server performs full transaction monitoring. IP Service Activator flow-through applications can therefore reliably track the progress of all IP Service Activator transactions as the corresponding configuration changes are activated on the network. This allows flow-through applications, which generally use IP Service Activator through its OIM APIs, to monitor the success or failure of their IP Service Activator transactions.

The IP Service Activator object model uses the concrete activation status to indicate whether a transaction's configuration changes, corresponding to each individual concrete, were successfully activated on the network.

The concrete activation status is recorded for the concretes affected by a particular transaction. It is aggregated into the overall provisioning status for the transaction. Both are attributes of the transaction object and can be observed in the IP Service Activator client and the APIs of the OSS Integration Manager:

- CORBA
- OSS Java Development Library (OJDL)

For more information about OJDL, see "The OSS Java Development Library".

More information about transaction monitoring functions is provided in *IP Service Activator System Administrator's Guide*.

Other Uses of OIM

Other software applications can use the OIM API to create and modify services for particular customers. For example, external applications can:

- Discover network elements, including details of interfaces, sub-interfaces and PVCs and their capabilities, or create devices for pre-provisioning.

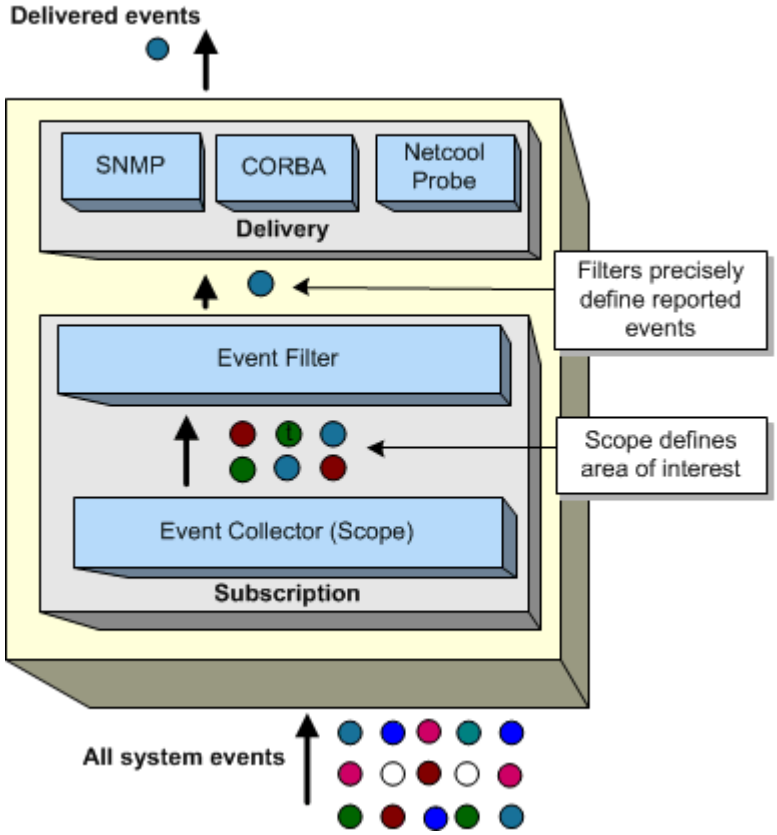
- Set up customers and sites, link sites to network elements and create VPNs and Transparent LAN Services.
- Set up and apply QoS or access control policies and apply them to specific points in the network topology.
- Subscribe to fault and state changes in any object.

Fault and Event Reporting

IP Service Activator can inform external applications of events or faults that occur throughout the network it is managing. This functionality is provided by the Event Handler, an optional component that is accessible through both OIM and the IP Service Activator client.

Figure 7–2 illustrates how the Event Handler filters events.

Figure 7–2 Event Handler Subscription Model



The Event Handler enables configurable, software-driven control of event reporting. It includes the following features:

- The Event Handler can report various events occurring in the managed network – the occurrence of a fault reported by IP Service Activator, the creation, deletion, linking or unlinking of an object, the change in the status of an object or the change in an attribute of an object.
- Users can define the events they are interested in by setting up subscriptions, which can be viewed, added, deleted and modified through the OIM or the IP Service Activator client.

- Events can be delivered using different methods (SNMP or CORBA).
- User-defined filters can enable subscribers to define exactly the events of interest, for example, specific faults and fault categories.

About Subscriptions

External applications communicate with the Event Handler by means of event subscriptions, which allow users of external applications considerable control over the events reported. For example, you can set up subscriptions to report on the following:

- All events and faults occurring anywhere in the system
- Faults reported in IP Service Activator components
- All events occurring in a specific VPN
- Events occurring on all PE devices in a network
- A set of specific faults relating to communication problems

Collecting and Filtering

The Event Handler gives subscribers considerable control over the events to be reported.

The primary point of interest can be defined. For example, an associated object in the network topology can be selected, such as a network or specific device. You can choose to report on this object only, or the object itself and all objects below it in the hierarchy. The scope can further be restricted by class of object, such as all devices, or all sites in a VPN. If you are reporting on devices and interfaces, you can choose to report only on those with a specific role. So, for example, you can set up a subscription to report on all configuration faults occurring on PE devices and their associated interfaces in VPNs associated with a specific customer.

For faults and attribute changes, you can apply filters to identify even more precisely the events to be reported. For faults, you can set up a filter to report faults in a particular category, or even by individual fault code. For attributes, you can select the individual object attribute to be monitored.

This categorization enables different types of messages to be accessed by different systems. For example, problems with provisioning may be of interest to billing systems, since requested services may have been compromised, while communication faults are more relevant to network engineers

Delivery Methods

Events can be delivered to external applications as SNMP traps by a CORBA interface.

SNMP Trap Reporting

Faults and clears reported from the IP Service Activator components can be delivered to other applications in the form of SNMP traps.

- A basic method, ensuring compatibility with earlier releases of IP Service Activator, reports the ID, severity, and description of the fault.
- An upgraded method enables events other than faults to be reported, and provides more information about the event, such as the IP address and status of an affected device.

Traps can be reported in SNMP v1 or v2 format.

CORBA Interface

The CORBA delivery mechanism allows CORBA clients to register and receive events by CORBA calls. The events are sent using the same format as defined by the CosNotification standard for structured events. An event channel name is defined in the IP Service Activator user interface (or through the OIM) when setting up the subscription object. A client may connect to one specific channel or all of the event channels defined in the subscriptions.

The OSS Java Development Library

The OSS Java Development Library (OJDL) is a toolkit that enables the development of web-based user interfaces to IP Service Activator and the integration of data from billing, fault or performance measurement software. Based on the OIM API, OJDL provides a set of Java classes that provide general access to the OIM and Java server pages ready for web deployment. An example web-based user interface is provided, which integrators can use as a basis for their own development.

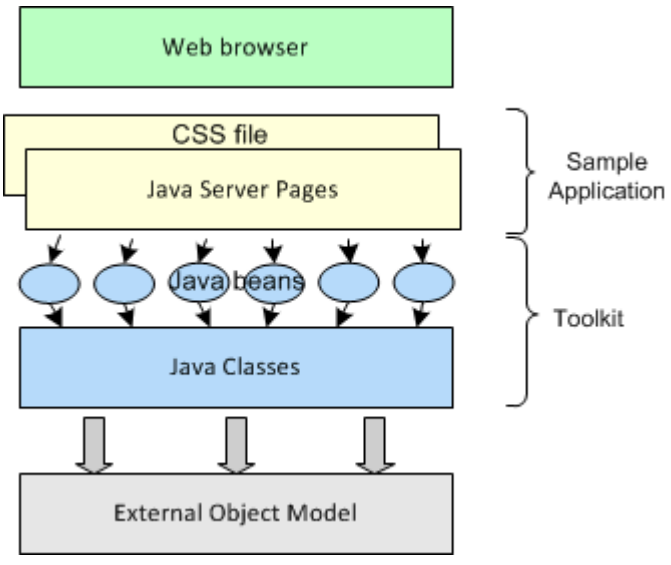
These features enable Service Providers to offer self-provisioning web interfaces, allowing their customers to order, view and amend their own services without any intervention by the provider. The interface can also be optimized for web self-management or centralized workflow provisioning, for example, for call center usage.

The OJDL consists of the following components:

- A toolkit: A set of Java classes and beans that may be used to integrate third-party applications or develop a Java front end to IP Service Activator.
- A sample interface: An example interface that may be used for demonstration purposes or as the basis of your own development.

The elements of the OJDL Toolkit and sample application are shown in Figure 7–3.

Figure 7–3 OJDL Components



The OJDL interfaces with IP Service Activator through the EOM. Objects in the EOM can be manipulated using pre-defined Java classes. These classes provide a generic front end to the OIM, supporting quick and easy integration by a Java developer. In addition, pre-defined Java beans simplify the development process, providing

reusable collections of the OIM commands required to perform a particular task, such as logging in.

Ready-to-Use Integrations

This section describes the ready-to-use integrations that come with IP Service Activator.

SLA Monitoring

The ability to offer and monitor Service Level Agreements is essential to service providers, who risk substantial losses if network downtime occurs and agreements are violated. Real-time and historical reports prevent network downtime by highlighting potential problems before they occur.

Ready-to-use integration reduces management and operational costs by providing automated activation and workflow management for the delivery of IP services and SLA reporting, including Service Assurance Agent (SAA) and NetFlow configuration.

Types of Measurements Supported

A number of range of measurement techniques are supported, enabling measurement of point-to-point performance, or performance at a specific point in the network:

- **Service Assurance Agent (SAA)**: A Cisco technology that performs point-to-point measurement based on key SLA metrics such as response time, network resources, availability, jitter, connect time and packet loss.
- **NetFlow**: A Cisco technology that enables you to characterize and analyze an IP flow on an interface. It is often used as a metering base for other applications, including accounting/billing, network planning, and marketing.
- **MIB2 statistics**: Report on Management Information Bases (MIBs) in MIB2 format. Their variables indicate the basic state of the network – for example, load, availability, discards and broadcast rate.
- **Class-based QoS MIB (CB QoS MIB) statistics**: Provide information about the policy and class maps that are configured on a device. In IP Service Activator terms, these statistics map onto the Class-based WFQ, WRED or MQC PHB groups that have been configured through the user interface.

Web Services and Reference Implementation

The web services API is an optional component of the IP Service Activator installation that allows IP Service Activator to integrate with Order and Service Management (OSM). For more information about using web services, see *IP Service Activator API Developer's Guide*.

Reference Implementation

This solution provides a reference implementation for IP Service Activator and OSM integration. Using the reference implementation, you can develop, integrate, and deploy Order and Service Management and IP Service Activator for flow-through activation of IP-related services.

The reference implementation includes the following:

- An Eclipse project that contains OSM workflows
- A Solution Uptake Guide

- Sample data to implement out-of-the-box workflows
- Set-up scripts that perform additional system configuration required for the reference implementation to run

For information about using best practices for reference implementations, see OSS Integrated Service Provisioning Reference Implementation for Layer 3 MPLS VPN Service with Metro Ethernet Access (Doc ID 1479821.1) knowledge article on Oracle Support.