Oracle® Communications IP Service Activator

OSS Integration Manager Guide

Release 7.2

E36051-01

October 2013

ORACLE®

Oracle Communications IP Service Activator OSS Integration Manager Guide, Release 7.2

E36051-01

Contents

3 The Command Language

4 The External Object Model

5 Examples of Using OIM

Preface

The Oracle Communications IP Service Activator *OSS Integration Manager Guide* (OIM) provides an outline for installing OIM, running OIM, integrating IP Service Activator web services with Order and Service Management (OSM), formatting of command languages. It includes reference information for the External Object Model, OIM examples, error handling, command grammar, command correlation, and a sample of IP Service Activator Transaction Publisher Class.

Audience

The *OSS Integration Manager Guide* is for designers writing interfaces to IP Service Activator. It provides details of the API, including command syntax and detailed examples.

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 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.

IP Service Activator Configuration Policy Online Documentation

You can access an online reference that documents configuration policies that are provided with IP Service Activator. This reference is distributed with the IP Service Activator Software Development Kit.

Access it using the following path:

*Service_Activator_home***ipsaSDK\doc\schemaDoc\configPolicyIndex.html**

Refer to the *Software Development Kit Installation and Setup Guide* for installation instructions and more information.

1

Introduction to the OSS Integration Manager

This chapter provides a brief introduction to the Oracle Communications IP Service Activator OSS Integration Manager (OIM) and its capabilities. It includes the following:

- A summary of what you can do with the OIM, and some security restrictions
- The OIM executables; how to install them and run them
- An introduction to the command set and the External Object Model
- An outline of connecting an OIM client to the Naming Service

What You Can Do with the OIM

The OIM provides an API to Operational Support Systems (OSSs), enabling IP Service Activator to be integrated with third-party software, such as billing, monitoring and fault management systems. The OIM enables automated or programmatic control of IP Service Activator's service provisioning facility.

Using OIM you can:

- Browse the managed network
- Provision services
- Report on faults and other events

You can also integrate IP Service Activator with Oracle Communications Order and Service Management (OSM) using the Web Services API. For information about using Web services with IP Service Activator, see "Web Services".

Browsing the Managed Network

A set of commands allows external systems to access part of the IP Service Activator object model. You can:

- Examine managed networks and VPNs
- View details of the services applied
- Check the status of managed devices or configured services
- Find objects by name or attribute

Provisioning Services

The OIM allows user applications to make both physical and logical changes to the topology. You can:

- Create or discover new devices, including details of interfaces, sub-interfaces and PVCs
- Set up customers and sites, link sites to network elements and create VPNs
- Create policy rules to control Quality of Service (QoS) and to control access, and apply them to specific points in the network topology
- Create standard and MQC Per Hop Behavior (PHB) groups and apply them to specific points in the network topology in order to control queuing or traffic shaping mechanisms
- Change certain attributes associated with a system, topology or policy object
- Propagate changes to the network, that is, update all devices/interfaces with configuration changes that have been specified
- Undo specific changes where you have ownership of the changes

Event Reporting

Faults and events occurring in any part of the managed network can be reported to external systems. By using the OIM in conjunction with the Event Handler, you can:

- Subscribe to specific types of events such as faults, system messages, status changes or changes in the attributes of objects
- Specify the parts of the network to be monitored, such as a specific VPN or subnet
- Deliver data in the form of SNMP traps, as CORBA calls or to Micromuse Netcool

For more detailed information about the Event Handler, see *IP Service Activator Event Handler Guide*.

Security Considerations

Use of the OIM requires a login and password, which must be set up from the IP Service Activator user interface. The user's rights to view, create, and modify objects are controlled by the user group and the permissions set up on objects. For information, see *IP Service Activator System Administrator's Guide.*

In addition, to ensure system security, some actions and some parts of the object model are not available using the OIM. You cannot:

- Access individual IP Service Activator system components, or execute core system operation commands, such as shutdown
- Change or alter the visibility of audit logs

Multiple Client OIM Access

IP Service Activator supports multiple OIMs running on one IP Service Activator system. You can connect multiple clients to each OIM instance.

Each OIM instance runs a copy of the master object model. Each client that is connected to an OIM instance can traverse and read this model concurrently with other clients. If a client performs any write operation on the model (i.e. create/modify/delete/use/unuse type commands, as opposed to setpath/getpath/getattributes/getparents type commands), these operations are stored in the OIM server in a transaction queue for that particular client.

At any point, the client can either issue an abort (which empties the queue) or a commit (which performs the queued operations on the model). If the client disconnects, this is treated as an abort (after an appropriate timeout).

When the queued operations begin to be applied to the model, it is locked for all other clients. Should other clients attempt to access the model, they will block. This situation resolves in one of two ways:

- The transaction is successfully applied into the model, model validation succeeds and the changes are successfully registered with the main object model in the policy server.
- One of the above steps triggers or encounters an error. The client is notified of transaction success/failure and the model re-opened to all clients.

Support for Load Balancing in Layer 3 VPNs

IP Service Activator has been enhanced to support the multipath statement for the Juniper Device Driver. Three new fields namely, Multipath, VpnUnequalCost, and EqualExternalInternal have been added to support load balancing in Layer 3 VPNs. This feature is supported only on Juniper Device Driver.

Support for 4-byte Autonomous System Number (ASN)

Autonomous System Number (ASN) is a unique number, which is used to identify Autonomous Systems under BGP routing. IP Service Activator 7.0.0 supports 4-byte ASN, an enhancement over the 2-byte ASN it supported in earlier versions. ASNs are used in generating various router commands under BGP and EBGP. IP Service Activator 7.0.0 supports 4 byte ASN to activate services on network elements.

In OIM, with the 4-byte support, the user can set 4-byte ASN values for most of the ASN fields like BGP ASN and site BGP ASN. However, for Route Distinguisher (RD) and Route Targets (RT), the user can enter both IP address and 4-byte ASN values.

Installing the OIM

The OIM is an optional IP Service Activator component, which consists of the following:

- The Integration Manager. The executable (**integration_manager.exe**) is installed in the **Program** directory on Windows systems and the **rbin** directory on Solaris systems
- The OIM command line interface (CLI). This optional executable (**integration_manager.cli**) allows you to run the OIM commands interactively.
- The OIM IDL file. This file (i**ntegration_manager.idl**) defines the OIM module structure in IDL (Interface Definition Language) format. This file is located in the **src** directory on Windows systems and on Solaris systems.

Note that a Component Manager must also be installed on the same host system.

> **Note:** Note that if you install the OIM on a Windows platform the CLI is only installed if you select the Extra Applications component.

For full installation details, see *IP Service Activator Installation Guide*.

Running the OIM

This section describes how to run the OIM.

Running the Integration Manager

To specify command-line parameters to OIM you need to do the following:

- On Solaris platforms, edit the Integration Manager entry in the configuration file **cman.cfg** in the IP Service Activator **Config** directory.
- On Windows platform you can use the Configuration GUI tool to run the Integration Manager by following the steps given below:
  - Ensure that you have installed OSS Integration.
  - Check the **OSS Integration Manager** check box in the Configuration GUI window, in the Config Parameters panel, under Component Manager Entries.
  - Click **Commit to host**.

> **Note:** For more information about the Configuration GUI, see *IP Service Activator System Administrator's Guide*.

Running the Command-line Interface

While generally applications are written in the form of scripts, the command-line interface is provided to run OIM commands interactively, which can be useful for testing purposes.

Run the following from the **Program** directory:

```
integration_manager_cli.exe
```

Table 1–1 lists and describes the command line parameters that can be set for the CLI.

Table 1–1 Command Line Parameters for CLI

Parameter	Description
-NoLogin	Use this option to prevent the login prompt. This option is useful if the CLI is to be used to run scripts.
-ComponentName *filename*	Use this option to specify the OIM component name if it is anything other than "**master_integration_manager**".
-ComponentLocation *hostname*	The name of the host on which the OIM is installed.
-ORBgiopMaxMsgSize size	Maximum CORBA message size in bytes. Defaults to 4294967295. See "Maximum CORBA Message Size".

You can create a simple text file of the OIM commands, then redirect the script into the CLI by using the following syntax:

```
integration_manager_cli -NoLogin < script.txt
```

In this case, the first line of the script should be:

```
login name=user_name password=password
```

Running OIM on Solaris

- Run the ./integration_manager_cli command in the ServiceActivator BIN directory. The path to that directory is **/opt/OracleCommunications/ServiceActivator/bin**.
- You are prompted for a login and password. The login and password are the same as your IP Service Activator login and password.

After you enter the login information, the Integration Manager prompt appears.

Maximum CORBA Message Size

In some situations, such as when an XML export is performed or when working with a large database, the default CORBA message size is not large enough. A message that exceeds the CORBA message size causes the OIM to fail. We recommend that any CORBA connection is initialized with the value 4294967295. This applies to both the connection between the OIM component and the policy server, and between an OIM client (such as the OIM command-line interface) and the OIM component. To initialize the OIM component and the OIM command-line interface with the required message size, use the command-line parameter:

```
-ORBgiopMaxMsgSize 4294967295
```

Any script that runs as a client of the OIM component should also initialize the CORBA connection with this recommended message size. The method used is dependent on the language in which the script is written.

Using the OIM

In order to write applications to interface to IP Service Activator, systems developers need the following:

- Details of the OIM command set and syntax
- Details of the IP Service Activator objects that can be viewed, created and modified via the API

The Command Set

The OIM command set comprises a number of generic commands that enable third-party systems to browse the object model and provision services. Full details are given in "The Command Language".

OIM uses a small set of generic commands, rather than a large number of commands specific to objects. This means the command set is independent of the IP Service Activator object model as well as any information model used by an external system. Changes can be made to object attributes or new objects added to the model without affecting backwards compatibility.

Any user familiar with UNIX or scripting language commands will find the OIM straightforward to use. All commands have the same basic format:

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 user interface. This logical approach helps ensure straightforward scripting.

The External Object Model

IP Service Activator maintains an Object Model of the physical managed network. This object model is managed by the policy server and stored in the central database. It holds information about all classes of objects, including their attributes, the actions that can be performed on them and the relationships between them. The object model is also known as the common object model, to distinguish it from local data models maintained on host machines running the user interface.

The External Object Model (EOM) is a subset of the common object model, including those objects which are accessible from external systems via the API.

Like the common object model, the EOM is divided into three major categories:

- **Topology**: the Topology Model contains objects that represent the physical network, such as VPNs, devices, interfaces and PVC objects.
- **Policy**: the Policy Model contains objects for creating and applying policies and services to the network devices.
- **System**: the System Model contains objects that represent the IP Service Activator system components and associated system management objects.

Full details are given in "The External Object Model".

Connecting an OIM Client to the Naming Service

The following is an example of Java code to connect to the Naming Service:

```
// setting up the CORBA details
org.omg.CORBA.Object objRef = orc.resolve_initial_references("NameService");
NamingContext nameServiceContext = NamingContextHelper.narrow(objRef);
NameComponent nc0 = new NameComponent("orchestream.com", "vendor");
NameComponent nc1 = new NameComponent("provider",    "application");
NameComponent nc2 = new NameComponent("integration_manager",   "component_type");
NameComponent nc3 = new NameComponent("master_integration_manager", "integration_
manager");

NameComponent path[] = {nc0, nc1, nc2, nc3};

org.omg.CORBA.Object sessionManagerRef = nameServiceContext.resolve(path);
OimSessionManager sessionManager =
OimSessionManagerHelper.narrow(sessionManagerRef);

org.omg.CORBA.Object sessionRef = sessionManager.NewSession("Java");
OimSessionInstance session = OimSessionInstanceHelper.narrow(sessionRef);
```

The complete path of the Integration Manager component in the Naming Service is:

```
 "orchestream.com", "vendor"
 "provider",        "application"
 "integration_manager","component_type"
 "<ComponentName>", "integration_manager"
```

2

Web Services

This chapter describes the Oracle Communications IP Service Activator Web services, which can be used to integrate IP Service Activator with Oracle Communications Order and Service Management (OSM).

About Web Services

IP Service Activator provides a Web services interface through which OSM can manage service activation transactions. For more information about OSM, see *OSM Concepts*.

Web services for IP Service Activator is an optional component that is available during IP Service Activator installation. For information about installing and configuring Web services, see *IP Service Activator Installation Guide*.

Web services is deployed on WebLogic Server. For information about WebLogic Server, see WebLogic Server documentation.

The Web service is an OSS/J-based interface that provides an external API for system integration. IP Service Activator transactions can be managed by the Web service. Each IP Service Activator request that is sent to the Web service contains a list of commands. These commands are then performed using a single transaction, without the need to specify the beginning and ending of the transaction. The Web service monitors these transactions and provides status notifications based on the result.

The external transport protocols for using Web services are HTTP, HTTPS, and JMS, and the data service formats are SOAP v1.1 and 1.2. Access-level security is provided through the implementation of the WebLogic WS-Policy specification, enforcing authentication. For more information about using SOAP as the data provider, see "OSM Data Providers".

About Web Services and OIM

The IP Service Activator Web service supports the use of multiple OSS Integration Manager (OIM) instances connected to a single instance of IP Service Activator. Use multiple OIMs to improve performance by allowing operations to be directed to OIMs on the basis of operation type, and also to allow for load sharing.

You can configure Web services to use previously configured OIMs using the Web Services configuration GUI. For information about configuring Web services at installation, see *IP Service Activator Installation Guide*. For information about the Configuration GUI, see *IP Service Activator System Administrator's Guide*.

Design Studio for IP Service Activator

An IP Service Activator activation cartridge is provided with the installation. This cartridge includes service action definitions that map to all IP Service Activator operations that are supported by the IP Service Activator Web services interface.

About Activation Tasks

Activation tasks within Design Studio provide integration between OSM and IP Service Activator. You must install OSM, IP Service Activator, and the IP Service Activator integration cartridge in Design Studio.

For information about working with activation tasks, see Design Studio online Help.

Configuring Web Services

Use the IP Service Activator configuration GUI to configure the Web service and deployment parameters, and to deploy or undeploy the Web service. For information about Web service parameters, see *IP Service Activator Installation Guide*, Post-Installation Tasks.

> **Note:** When you change IP Service Activator Web service parameters, re-deploy the Web service to ensure that the changes take effect.

For information about configuring other components in the configuration GUI, see *IP Service Activator System Administrator's Guide.*

Security

IP Service Activator Web Service access control security determines the functionality that each user has access to. To set up access control security, create a security role. Give this role the privilege to start IP Service Activator Web Services. When the Web service client needs to access the Web service, the client needs to authenticate itself to the Oracle WebLogic Server hosting IP Service Activator Web Service. See *Oracle WebLogic Administration Guide* for information about setting up access security.

> **Note:** Oracle WebLogic access control security protects only WebLogic Server resources and does not cover secure communication with IP Service Activator Web Services. As a result, SOAP messages transmitted between the Web service and its clients are in plain text.

Web Services allows only access level security. Clients must use a user id that is a member of group IPSA_WS_USERS_GROUP to communicate with IP Service Activator Web Services. The web.xml file defines the security role IPSA_WS_USERS and weblogic.xml file defines the security principal name as IPSA_WS_USERS_GROUP.

The installer creates a default user. For information about the default user names and passwords used with Web services, see *IP Service Activator Installation Guide*. This user is a member of the IPSA_WS_USERS_GROUP group. Due to limitations of the WebLogic console, information created by the command line tools, such as the role name, might not be available on the console.

OSM Data Providers

A data provider is used to retrieve data in an XML format from external systems.

The data provider type for IP Service Activator Web service is SOAP, which lets you access Web services from OSM and use the responses within behaviors. The required parameter is **soap.endpoint**, which is an element that identifies the URL to which the SOAP request is sent. Specify the **soap.endpoint** as:

- **http://***ws_IP:Port***/Oracle/IPSAView/Ws/Http**

Where *ws_IP:Port* is the IP Service Activator Web service IP and port.

The **soap.action** parameter contains the URI that identifies the intent of the message. There are several actions that you can specify in the **soap.action** parameter for IP Service Activator Web services. Table 2–1 lists these actions and their corresponding parameters. For more information about the built-in SOAP data provider parameters, and about using data instance behaviors, see *Order and Service Management Developer's Guide*.

Table 2–1 Action Parameters

soap.action	Parameters
tns/getObject	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getObject><-- Mandatory. Path to an object --> </ws:getObject> </soap:Body>
tns/getId	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getId><-- Mandatory. Path to an object --> </ws:getId> </soap:Body>
tns/getPath	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getPath><-- Mandatory. Object ID whose path is required --> </ws:getPath> </soap:Body>
tns/getName	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getName><-- Mandatory. Path to an object or its ID --> </ws:getName> </soap:Body>
tns/getParents	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getParents><-- Mandatory. Path to an object or its ID --> </ws:getParents> </soap:Body>

Table 2–1 (Cont.) Action Parameters

soap.action	Parameters
tns/getChildren	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getChildren><-- Mandatory. Path to an object or its ID --> </ws:getChildren> </soap:Body>
tns/getAttributes	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getAttributes> <-- Mandatory. Path to an object or its ID --> </ws:getAttributes> </soap:Body>
tns/getTransactionStatus	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws: getTransactionStatus> <-- Mandatory. Path to an object or its ID --> </ws: getTransactionStatus> </soap:Body>
tns/getTargets	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:getTargets> <-- Mandatory. Path to a ParameterSetInstance object or its ID --> </ws:getTargets> </soap:Body>

Table 2–1 (Cont.) Action Parameters

soap.action	Parameters
tns/find	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:find> <-- Mandatory. Path to an object or its ID. This is the start point of the search --> <ws:objectId></ws:objectId> <-- Mandatory. Type of object to find such as Device, Interface, etc. --> <ws:type>Interface</ws:type> <-- Mandatory. Name of the object to find. --> <-- Wildcards are allowed: * (any number of characters), or ? (any single character) --> <ws:name>Serial*</ws:name> <-- Optional. Indicates the direction of the search. Possible values are: parent, child. --> <-- parent - to search upwards through the hierarchy. --> <-- child - to search downwards through the hierarchy (this is the default) --> <ws:findDirection>child</ws:findDirection> <-- Optional. attribute=value pairs on which to search. --> <-- Wildcards are allowed in string arguments: * (any number of characters), or ? (any single character) --> <ws:attr1Name></ws:attr1Name> <ws:attr1Value></ws:attr1Value> <ws:attr2Name></ws:attr2Name> <ws:attr2Value></ws:attr2Value> <ws:attr3Name></ws:attr3Name> <ws:attr3Value></ws:attr3Value> <ws:attr4Name></ws:attr4Name> <ws:attr4Value></ws:attr4Value> <ws:attr5Name></ws:attr5Name> <ws:attr5Value></ws:attr5Value> </ws:find> </soap:Body>

Table 2–1 (Cont.) Action Parameters

soap.action	Parameters
tns/findParameters	<soap:Body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws"> <ws:findParameters> <-- Mandatory. Path to an object or its ID. This is the start point of the search --> <ws:objectId><ws:objectId> <-- Mandatory. Type of object to find such as Device, Interface, etc. --> <ws:type>Interface</ws:type> <-- Mandatory. Name of the object to find. --> <-- Wildcards are allowed: * (any number of characters), or ? (any single character) --> <ws:name>Serial*</ws:name> <-- Optional. Indicates the direction of the search. Possible values are: parent, child. --> <-- parent - to search upwards through the hierarchy. --> <-- child - to search downwards through the hierarchy (this is the default) -->

Finding and Retrieving Data

Using parameters for search criteria, you can find and retrieve data, such as objects, in IP Service Activator. When searching, the supported wildcard characters are "*" and "?".

For more information, see *Order and Service Management Developer's Guide*.

Finding Objects

Find objects parameters allow you to find objects in IP Service Activator. For information about objects in IP Service Activator, see *OSS Integration Manager Guide*.

Use the find objects parameters listed in Table 2–2 to find IP Service Activator objects.

Table 2–2 Find Objects Parameters

Find Objects Parameters	Definition
parentID	The starting path for the IP Service Activator FIND command.
type	The type of object to find in IP Service Activator.
name	The name of the object to find in IP Service Activator.
direction	Search direction; either parent or child.
attr1Name	The first attribute name.
attr1Value	The first attribute value.
attr2Name	The second attribute name.
attr2Value	The second attribute value.

Table 2–2 (Cont.) Find Objects Parameters

Find Objects Parameters	Definition
attr3Name	The third attribute name.
attr3Value	The third attribute value.
attr4Name	The fourth attribute name.
attr4Value	The fourth attribute value.
attr5Name	The fifth attribute name.
attr5Value	The fifth attribute value.

The following is an example service request:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header />
    <env:Body>
    <find xmlns="http://www.oracle.com/cgbu/ipsa/osmipsa/ws">
      <parentId>/Policy:"policy"</parentId>
      <type>Concreteobject</type>
      <name>*</name>
      <direction>child</direction>
      <!--Optional:-->
      <att1Name>state</att1Name>
      <!--Optional:-->
      <att1Value>installed</att1Value>
      <!--Optional:-->
      <att2Name />
      <!--Optional:-->
      <att2Value />
      <!--Optional:-->
      <att3Name />
      <!--Optional:-->
      <att3Value />
      <!--Optional:-->
      <att4Name />
      <!--Optional:-->
      <att4Value />
      <!--Optional:-->
      <att5Name />
      <!--Optional:-->
      <att5Value />
    </find>
  </env:Body>
  </env:Envelope>
  Service Response
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header />
    <env:Body>
    <ws:findResponse xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws">
      <ws:eomObject>
        <ws:objectId>1425</ws:objectId>
        <ws:objectName>1111 on Serial2/0/1.1/1/1/1:2</ws:objectName>
        <ws:objectType>ConcreteObject</ws:objectType>
      </ws:eomObject>
      <ws:eomObject>
        <ws:objectId>1430</ws:objectId>
        <ws:objectName>111 on Serial2/0/1.1/1/1/1:2</ws:objectName>
        <ws:objectType>ConcreteObject</ws:objectType>
```

```
      </ws:eomObject>
    </ws:findResponse>
  </env:Body>
  </env:Envelope>
```

Retrieving Objects

Use find parameters to retrieve objects in IP Service Activator.

Use the parameters listed in Table 2–3 to retrieve IP Service Activator objects to which a ParameterSetInstance object applies.

Table 2–3 Find Parameters for Retrieving Objects

findParameter	Definition
parentID	The starting path for the IP Service Activator FIND command.
type	The type of object to find in IP Service Activator.
direction	Search direction; either parent or child.
attr1Name	The first attribute name.
attr1Value	The first attribute value.
attr2Name	The second attribute name.
attr2Value	The second attribute value.
attr3Name	The third attribute name.
attr3Value	The third attribute value.
attr4Name	The fourth attribute name.
attr4Value	The fourth attribute value.
attr5Name	The fifth attribute name.
attr5Value	The fifth attribute value.

The following is an example retrieval request:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header />
    <env:Body>
    <findParameters xmlns="http://www.oracle.com/cgbu/ipsa/osmipsa/ws">
      <parentId>/</parentId>
      <type>devicetype</type>
      <direction>child</direction>
      <!--Optional:-->
      <att1Name />
      <!--Optional:-->
      <att1Value />
      <!--Optional:-->
      <att2Name />
      <!--Optional:-->
      <att2Value />
      <!--Optional:-->
      <att3Name />
      <!--Optional:-->
      <att3Value />
      <!--Optional:-->
      <att4Name />
      <!--Optional:-->
```

```
      <att4Value />
      <!--Optional:-->
      <att5Name />
      <!--Optional:-->
      <att5Value />
    </findParameters>
  </env:Body>
  </env:Envelope>
  Service Response
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header />
    <env:Body>
    <ws:findParametersResponse
xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws">
      <ws:objects>
        <ws:objectId>551</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine 80E</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>552</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine 5000E</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>553</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine NE5000EMulti</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>554</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine NE40E</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>555</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine NE20E-4</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>556</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine NE20E-8</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>557</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine 40-4</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>558</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine 40-2</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
        <ws:objectId>559</ws:objectId>
        <ws:objectName>Huawei Quidway NetEngine 40-8</ws:objectName>
        <ws:objectType>DeviceType</ws:objectType>
      </ws:objects>
      <ws:objects>
```

```
      <ws:objectId>560</ws:objectId>
      <ws:objectName>Huawei Quidway Eudemon 500</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>561</ws:objectId>
      <ws:objectName>Huawei Quidway Eudemon 1000</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>562</ws:objectId>
      <ws:objectName>Foundry NetIron IMR 640</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>563</ws:objectId>
      <ws:objectName>Foundry NetIron XMR 4000</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>564</ws:objectId>
      <ws:objectName>Foundry NetIron XMR 8000</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>565</ws:objectId>
      <ws:objectName>Foundry NetIron XMR 16000</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>566</ws:objectId>
      <ws:objectName>Foundry NetIron MLX-4</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>567</ws:objectId>
      <ws:objectName>Foundry NetIron MLX-8</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>568</ws:objectId>
      <ws:objectName>Foundry NetIron MLX-16</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>569</ws:objectId>
      <ws:objectName>Paradyne GranDSLAM 4200</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>570</ws:objectId>
      <ws:objectName>RedBack Networks SMS 1000</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
    <ws:objects>
      <ws:objectId>571</ws:objectId>
      <ws:objectName>RedBack Networks SMS 500</ws:objectName>
      <ws:objectType>DeviceType</ws:objectType>
    </ws:objects>
  </ws:findParametersResponse>
```

```
</env:Body>
</env:Envelope>
```

Retrieving Other Data

You can also use parameters to retrieve other data in IP Service Activator, for example, attributes, child objects, and immediate parent objects.

Use the parameters in Table 2–4 to retrieve other data in IP Service Activator.

Table 2–4 Find Parameters for Retrieving Other Data

findParameter	Definition
getAttributes	Retrieves attributes of an object. Path: the path/ID to the object.
getChildren	Retrieves immediate child objects of an object. Path: the path/ID to the objects.
getId	Retrieves the ID of an object. Path: the path/ID to the objects.
getName	Retrieves the name of an objects. Path: the path/ID to the objects.
getOrderStatus	Retrieves the order status of an OSM order. orderKey: the key of an OSM order.
getParents	Retrieves the immediate parent objects of an object. Path: the path/ID to the objects.
getPath	Retrieves the path on an object. Path: the path/ID to the object.
getTargets	Retrieves the targets that a ParameterSetInstance object applies to. Path: the path/ID to the object.

The following is an example for getAttributes:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header />
    <env:Body>
    <getAttributes
xmlns="http://www.oracle.com/cgbu/ipsa/osmipsa/ws">[13]</getAttributes>
  </env:Body>
  </env:Envelope>
  Service Response
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header />
    <env:Body>
    <ws:getAttributesResponse
xmlns:ws="http://www.oracle.com/cgbu/ipsa/osmipsa/ws">
      <ws:attribute>
        <ws:attributeName>name</ws:attributeName>
        <ws:attributeValue />
      </ws:attribute>
      <ws:attribute>
        <ws:attributeName>remarks</ws:attributeName>
        <ws:attributeValue />
```

```
    </ws:attribute>
    <ws:attribute>
      <ws:attributeName>type</ws:attributeName>
      <ws:attributeValue>Any</ws:attributeValue>
    </ws:attribute>
    <ws:attribute>
      <ws:attributeName>id</ws:attributeName>
      <ws:attributeValue>13</ws:attributeValue>
    </ws:attribute>
  </ws:getAttributesResponse>
</env:Body>
</env:Envelope>
```

Web Service Operations

The set of operations described in Table 2–5 is provided.

Table 2–5 Web Service Operations

Operation	Definition	Parameter
createOrderByValue	Converts an OSM order to an IP Service Activator transaction.	OrderValue
cancelOrderByKey	The Web service rolls back the corresponding IP Service Activator transaction.	OrderId
abortOrderByKey	The Web service rolls back the corresponding IP Service Activator transaction.	OrderKey

3

The Command Language

This chapter provides details of the commands available within Oracle Communication IP Service Activator OSS Integration Manager.

Command Grammar

All commands take the following basic form:

```
command [object-path] [attributes]
```

All commands operate on objects within the EOM, identified by a unique name or a unique ID number. For full details of the objects that can be accessed, see "The External Object Model".

See "Command Grammar" for a formal definition of the language grammar.

Object Path

The **object_path** identifies the object that the command is to operate on. Not all commands require an object path. If the command requires an object path and it is omitted then the current object is assumed as the path.

Note that the **link** and **unlink** commands operate on two objects, a parent and a child, so the paths of both objects must be specified.

The **object_path** can take one of three forms:

- An absolute object path, starting with the root object. The '/' character given at the start of the path indicates the root object. For example:

  ```
  /System:"System"/EventSubscription:"DeviceFaults"
  ```

- A relative object path from the current object. For example:

  ```
  network:"Acme"/device:"Cisco7505"
  ```

- An object's unique ID number, enclosed in square brackets. For example:

  ```
  [675]
  ```

Table 3–1 shows and defines the symbols that may be used in the path definition.

Table 3–1 Path Definition Symbols

Symbol	Description
.	Current path
..	Path to parent object

Table 3–1 (Cont.) Path Definition Symbols

Symbol	Description
/	Object separator
Object name	Set the path to the location of the specified object
Object ID	Set the path to the location of the object with the specified ID

Note: During object creation, unique object name verification is performed within its parent object. If duplicate names are discovered, an error message similar to the following will appear:

```
CommandExecutionError.ERR_ObjectModelViolation: Object [1802], a
Interface, has multiple children of type SubInterface with the same
name: Serial1/0.100, [4129].
```

You can create multiple objects with the same name in an object model as long as they belong to different parent objects. Uniquely named folders, under a parent object, can each store objects with the same name. The stored objects can have the same name since they belong to different parent objects

Attributes

Not all commands require attributes. When set, they take the form:

```
attribute="value"
```

Quotation marks are not strictly required unless there are spaces within the value. If necessary, multiple attributes can be specified on a single line.

Data Types

Attributes entered in command lines can be one of the data types displayed in Table 3–2.

Table 3–2 Data Types

Type	Description
String	A string containing any alphanumeric characters. Maximum string length is generally 127 characters; where the maximum length of a string is shorter or longer, its length is specified in this document.
Object ID	The unique ID of each object, in the form [*nnn*] where *nnn* is an unsigned integer (When attributes are returned, object IDs appear as they are in integer form)
DateTime	YYYY/MM/DD HH:MM:SS The date defaults to today
IpAddress	nnn.nnn.nnn.nnn where nnn <= 255
Boolean [string]	True or False

Table 3–2 (Cont.) Data Types

Type	Description
Enum	String representing one member from a set of possible enumerated values. Enum stands for enumerated type. Each attribute of type Enum is described with an enumeration of valid arguments. This attribute can be entered either by its numeric value or by the corresponding string.
U32	Unsigned 32-bit integer
Bitmap	Some attributes have binary values that are additive. When several choices in a range can be set to true, then the attribute value is a binary sum that uniquely represents those several choices.

Case Sensitivity

Commands and parameters are not case-sensitive, with the exception of user passwords which must be entered exactly as originally set.

Overview of the Commands

OIM commands are grouped into the following modules:

- Access Control Module commands: allow the user to securely authenticate to IP Service Activator
- Transaction Module commands: aggregate and execute changes to the database and network into transactions
- Declarative Module commands: modify, link, create, delete or otherwise change objects within the topology or policy tree
- Management Module commands: discover and manage the network
- Navigation Module commands: navigate or search the object model
- Utility Module commands: facilitate the use of OIM. Commands in the Utility Module are executed immediately

Transactions

Unless specified in their individual descriptions, commands that belong to the Declarative and Management modules are held in a default transaction queue. The commands in the transactions can be executed with a **commit** command, or the queue can be removed by an **abort** command. Commands held in the current transaction queue can be listed using the **listTrans** command.

Navigation commands may be interspersed with Declarative and Management commands. Navigation commands are executed immediately.

For more details about transaction processing within IP Service Activator, see *IP Service Activator Concepts Guide*.

Error Reporting

Details of errors are given in "Error Handling".

The Access Control Module

The Access Control Module commands allow the user to securely log into and authenticate to the system by providing a user name and a password. The user can exit by logging off at any time.

Users must already have been set up within IP Service Activator.

Table 3–3 summarizes the Access Control Module commands.

Table 3–3 Access Control Module Commands

Command	Summary
login	Logs the user into IP Service Activator
logout	Logs the user out of IP Service Activator
getAccess	Get the user's access rights for all object types or a specific object type

The Login Command

The **login** command logs the user into the system, using a log-in name and password previously set up in IP Service Activator. You can use the **login** command in scripts to execute commands on the OIM client without logging into it. The log in information is provided in these scripts.

Table 3–4 lists the attributes for the login command.

Table 3–4 login Command Attributes

Attribute	Type	Description
name	String	User name previously set up in IP Service Activator.
password	String	Password previously set up in IP Service Activator.

Syntax

The syntax for the **login** command is:

```
login name=username password=password
```

where *username* and *password* are the log-in name and password previously set up in IP Service Activator.

For example:

```
login name="user1" password="mypass26"
```

Path

[no default]

Errors

The **LoginException** error may be returned for the following reasons:

- The user has entered an invalid combination of username and password.
- The user has repeatedly failed to log into the system.
- Concurrent logins are not permitted.
- The password has expired.

The ChangePassword Command

The **changePassword** command enables the user to set or modify their own password.

Syntax

The syntax for the **changePassword** command is:

```
changepassword
```

The user is prompted to enter a password and to confirm it. Any characters entered are displayed on screen in encrypted mode.

For example:

```
changepassword
password: *******
confirmation password: *******
```

Path

[no default]

Return

[no default]

Errors

[no default]

The Logout Command

The **logout** command logs the user off the system.

Syntax

The syntax for the **logout** command is:

```
logout
```

Path

[no default]

Return

[no default]

Errors

[no default]

The getAccess Command

The **getAccess** command returns the current user's access permissions for a single object, an object type or all object types.

Table 3–5 lists the attributes for the getAccess command.

Table 3–5 getAccess Command Attributes

Attribute	Type	Description
type	String	Object type name, or "all" to retrieve the information for all object types.

Syntax

The syntax for the **getAccess** command is:

```
getAccess [object-path] [type=objecttype]
```

where *object-path* is the path to object for which access rights are required and *objecttype* is an object type name. Use **all** instead of an object type name to retreive information for all object types.

For example:

- To return access rights for the current object:

  ```
  getAccess
  ```

- To return access rights for the object with ID no 250:

  ```
  getAccess [250]
  ```

- To return access rights for all Device objects:

  ```
  getAccess type=Device
  ```

- To return access rights for all objects:

  ```
  getAccess type=all
  ```

Path

The object for which access rights are required. This is ignored if a type attribute is specified.

Return

The **getAccess** command returns a **True** or **False** value for each of the following access rights:

- read
- create (includes delete)
- link (includes unlink)
- modify
- execute

Errors

[no default]

The Transaction Module

The Transaction Module commands allow the user to save commands in the form of transactions and control how and when the transactions are executed.

Commands in the Declarative and Management modules are grouped into transactions. Other commands are executed immediately.

The Transaction Module provides the commands summarized in Table 3–6.

Table 3–6 Transaction Module Commands

Command	Summary
abort	Stops the current transaction.
commit	Saves the changes made in the current transaction to the database.
events	Returns the changes that have occurred through transactions since the last time the **events** command was called.
exporttransaction	Returns the changes that have occurred through transactions since the last time the **events** command was called.
listTrans	Lists commands in the current transaction.
merge	Applies changes saved in a Transaction object to the current object model.
rollback	Discards changes stored in a Transaction object in the object mode.
schedule	Schedules a Transaction object to be applied at a given date and time.

For more details about transaction processing see *IP Service Activator Concepts Guide*.

The Abort Command

The **abort** command stops the current transaction and discards the queued commands.

Syntax

The syntax for the abort command is:

```
abort
```

Path

[no default]

Return

[no default]

Errors

[no default]

The Commit Command

The **commit** command saves the changes within the current transaction in the object model, and sends them to the server.

A transaction can be:

- Committed immediately. The common object model is updated with the transaction's changes and any configuration changes are propagated to the network.
- Queued. The transaction is saved to the common object model's transaction store with a status of Pending. The object model changes associated with that transaction are not executed and no configuration changes are propagated to the network.

- Scheduled. The transaction is saved and will be committed at a specified future date or time.

In all cases, the **commit** command creates a TransactionEntry object.

Atomic transactions change the model for successful execution of the transaction. Normally, all commands in a transaction must be executed successfully for the transaction as a whole to be completed i.e. if any one command in a transaction fails, the entire transaction fails.

Use atomic transactions to break a transaction into smaller sub-transactions which can succeed or fail independently without causing the entire transaction to fail.

These sub-transactions are grouped using the commit type **atomic**. Then entire transaction, which includes the atomic sub-transactions, is then committed using the commit type **atomicTransaction**.

Notice the error in the first command of the following example:

```
create Customre:"Bob"
create Customer:"Bob"/Site:"NOC"
commit mode=atomic
create Customer:"John Doe"
create Customer:"John Doe"/Site:"NOC"
commit mode=atomic
commit mode=atomicTransaction
```

The first command fails, causing the first atomic transaction, which attempts to create customer Bob, to fail. However, the second atomic transaction to create customer John Doe and its NOC succeeds. The post transaction activities which apply to the second atomic transaction are still completed.

If a command in a transaction fails, and it isn't in an atomic transaction, the whole transaction fails, regardless of whether or not it contained atomic transactions.

Table 3–7 describes the attributes used by the **commit** command.

Table 3–7 Attributes Used by the Commit Command

Attribute	Type	Description
mode	Enum	The type of commit to perform. Values are: ■ immediate ■ queue ■ schedule ■ atomic ■ atomicTransaction The default value is immediate.
name	String	The name of the TransactionEntry object that is created. Optional for an immediate commit, mandatory for a queued or scheduled commit.
schedule	DateTime	The date and time that the transaction is to be applied. Mandatory if **mode** is schedule.
waitforcompletion	Boolean	When set to True, the **commit** command returns only if the transaction's changes have been written to the database. The default is False.

Syntax

The syntax for the **commit** command is:

```
commit [mode=commit-mode] [name=transaction-name] [schedule=datetime]
```

For example:

- To commit the current transaction:

  ```
  commit
  ```

- To schedule the current transaction to be executed at a specific time:

  ```
  commit mode=schedule name="Scheduled Transaction" schedule="2002/11/15
  20:00:00"
  ```

- To queue a current transaction:

  ```
  commit mode=queue name="createVPN"
  ```

Path

[no default]

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- Commit Failure: an element of the commit transaction was not successful
- No name attribute was specified to a queued or scheduled transaction
- No schedule time was specified for a scheduled transaction

The Events Command

The **events** command returns the changes that have occurred (through transactions) since the last time the **events** command was called.

The OIM starts to record events when the **events** command is first entered. When the **events** command is next called all the recorded events are returned.

The command can be used in two ways:

- If no path is specified, the command lists the changes that have occurred since the last time the command was called.
- If a path is specified, the command lists the transactions within the identified transaction object.

Table 3–8 lists the attributes for the events command.

Table 3–8 events Command Attributes

Attribute	Type	Description
block	Boolean	When set to **True**, if no changes have occurred since the last time the events command was called, the command pauses its execution until an event occurs. This eliminates the process of having to repeatedly enter the events command until an event occurs. This automatically returns details of the transaction that has occurred.

Syntax

The syntax for the **events** command is:

```
events [object-path] [block=true]
```

For example:

- To return information about a committed transaction:

  ```
  events [2037]
  ```

- To run the events command in blocking mode, in order to pause the execution of the command until an event occurs:

  ```
  events block=true
  ```

Path

The path to the TransactionEntry object. A path does not have to be specified.

Return

The **events** command returns the commands within the committed transaction.

Errors

- The **CommandExecutionError** may be returned if the object specified in the path is not a TransactionEntry object.
- The **OimSystemException** command may be returned if the OIM session is invalid, or if OIM is shutting down.

The Exporttransaction Command

The **exporttransaction** command outputs the list of operations part of the transaction. The output is formatted as follows for three formats of output: normal, concrete and fault operations:

```
OID: <main_object_ID>      OClass: <main_object_class_type>    OName: <main_
object_name> Operation: <type_of_operation>     PClass: <parent_object_class>
PName: <parent_object_name> PId: <parent_object_id>    Parameters:
<Attribute1=Value1>, <Attribute2=Value2>…
CID: <concrete_ID>      CClass: <Concrete_object_class_type>     Operation:
<type_of_operation>     PClass:      PName:  PId:    Parameters:
<Attibute1=Value1>…
Fault Operations:
FID: <object_ID>      Operation: <type_of_operation>       PClass:         PName:
PId:    Parameters: <Attribute1=Value1>, <Attribute2=Value2>…
```

> **Note:** PClass, PName, and PId are only used for "link" type of operations to describe the parent object to which the main object is to be linked as a result of the link operation.

Syntax

The syntax for the **exporttransaction** command is:

```
exporttransaction [TransactionEntry-object-id]
```

where *TransactionEntry-object-id* is the object ID for which the list of operations is to be displayed.

Path

[no default]

Return

The exporttransaction command returns a list of operations for the transaction.

Errors

- Transaction entry ID not specified:

  CommandExecutionError.ERR_ExportTransIdRequired: The command needs a transaction entry id.

- Transaction entry not found:

  CommandExecutionError.ERR_CannotResolveToObject: Cannot Resolve to object: [<object_id>]

Concretes Affected by the Exporttransaction Command

Concretes can be affected when a Policy Rule or Configuration Policy is changed as part of the Modify Operation of a transaction.

Possible values for OPERATION are:

- Create
- Link
- Unlink
- Modify
- Delete

For Object instances, concretes can be Linked, Unlinked, Affected, and Deleted in a transaction.

CID is the Concrete ID of the concretes that will be created when Policy Rules (Access Rule, Classification Rule, Configuration Policy etc.) or Configuration Policies (Banners, Static Routes etc.) are applied to Policy Targets (Device, Interface, VPN etc).

While displaying transaction details, renumbering of Concrete IDs may occur if a Policy Rule is applied to a Policy target in one transaction that results in creation of new concretes that are different from previously created (and currently non-existent) concretes when the same Policy Rule has been applied to the same Policy Target in another transaction.

Enhancement of Exporttransaction Command and the Cat Command

In the GUI, for a transaction entry object, viewing and exporting of concrete data is performed in two ways, namely raw data and modified data. In OIM, for **cat** and **exporttransaction** commands, a new argument showrawdata has been introduced . Depending on the value of showrawdata the concrete data is displayed. If showrawdata=true, the raw data is displayed and if showrawdata=false, the modified data is displayed. The display is consistent across GUI and OIM. This implies that the GUI and OIM by default display the modified data, whereby the showrawdata is always false.

Since **cat** is a very generic command, showrawdata is only applicable for transactionentry objects. If users try to use the showrawdata with other objects they will encounter an error. In the GUI, a new check box is added. If the user selects the

check box, the original concrete IDs are displayed. In the GUI, the check box option is named **Show original Concrete IDs**.

Exporttransaction Command Examples

The following example shows renumbered CIDs in the output of the **exporttransaction** command. The example shows that the Create and Link concretes are renumbered. You will notice that CID 1741 is renumbered to 1748 when AccessRule AR1 is applied to the Interface with OID 1685.

```
/>exporttransaction [1742]
Operations:
OID: 1739       OClass: AccessRule      OName: AR1      Operation: Create
PClass:         PName:  PId:    Parameters: AclName=, AclNumber=0, Conflict=0,
Disabled=0, Fragments=0, Inbound=1, Log=0, ManagementOverride=0, Name=AR1,
NamedAcl=1, Order=4026531840, Outbound=0, Permit=1

OID: 1739       OClass: AccessRule      OName: AR1      Operation: Link PClass:
Network PName: test     PId: 578        Parameters: OID: 1740       OClass:
ClassificationLeaf      OName:  Operation: Create       PClass:         PName:
PId:    Parameters: AclIdType=2, AclName=, AclNumber=0, AddressType=0,
Aggregate=1, DestinationIpAddr=0.0.0.0, DestinationIpv6Addr=, DestinationMacAddr=,
DestinationMacMask=, DestinationMask=0.0.0.0, Fragments=0, Log=0, Match=1,
MatchType=1, Name=, Remarks=, SourceIpAddr=0.0.0.0, SourceIpv6Addr=,
SourceMacAddr=, SourceMacMask=, SourceMask=0.0.0.0

OID: 1740       OClass: ClassificationLeaf      OName:  Operation: Link PClass:
AccessRule      PName: AR1      PId: 1739       Parameters: OID: 5  OClass:
DeviceRole     OName: 2       Operation: Link PClass: AccessRule     PName: AR1
PId: 1739       Parameters:
OID: 11 OClass: InterfaceRole   OName: 3      Operation: Link PClass: AccessRule
PName: AR1      PId: 1739       Parameters: Concrete Operations:

CID: 1741      CClass: ConcreteAccessRule     Operation: Create     PClass:
PName:  PId:    Parameters: AuditMismatchIgnored=0, AuditState=0, Conflict=0,
Direction=0, Enabled=1, NotificationCount=0, State=0
CID: 1741     CClass: ConcreteAccessRule     Operation: Link PClass: AccessRule
PName: AR1      PId: 1739       Parameters:

CID: 1741     CClass: ConcreteAccessRule     Operation: Link PClass: Interface
PName: 1685     PId: 1685       Parameters:

/>exporttransaction [1742]
Operations:
OID: 1739       OClass: AccessRule      OName: AR1      Operation: Create
PClass:         PName:  PId:    Parameters: AclName=, AclNumber=0, Conflict=0,
Disabled=0, Fragments=0, Inbound=1, Log=0, ManagementOverride=0, Name=AR1,
NamedAcl=1, Order=4026531840, Outbound=0, Permit=1

OID: 1739       OClass: AccessRule      OName: AR1      Operation: Link PClass:
Network PName: test     PId: 578        Parameters: OID: 1740       OClass:
ClassificationLeaf      OName:  Operation: Create       PClass:         PName:
PId:    Parameters: AclIdType=2, AclName=, AclNumber=0, AddressType=0,
Aggregate=1, DestinationIpAddr=0.0.0.0, DestinationIpv6Addr=, DestinationMacAddr=,
DestinationMacMask=, DestinationMask=0.0.0.0, Fragments=0, Log=0, Match=1,
MatchType=1, Name=, Remarks=, SourceIpAddr=0.0.0.0, SourceIpv6Addr=,
SourceMacAddr=, SourceMacMask=, SourceMask=0.0.0.0 OID: 1740       OClass:
ClassificationLeaf     OName:  Operation: Link PClass: AccessRule     PName: AR1
PId: 1739       Parameters:
```

```
OID: 5  OClass: DeviceRole     OName: 2       Operation: Link PClass: AccessRule
PName: AR1      PId: 1739       Parameters: OID: 11 OClass: InterfaceRole   OName:
3        Operation: Link PClass: AccessRule      PName: AR1      PId: 1739
Parameters:

Concrete Operations:
CID: 1748      CClass: ConcreteAccessRule     Operation: Create     PClass:
PName:  PId:    Parameters: AuditMismatchIgnored=0, AuditState=0, Conflict=0,
Direction=0, Enabled=1, NotificationCount=0, State=0

CID: 1748     CClass: ConcreteAccessRule     Operation: Link PClass: AccessRule
PName: AR1      PId: 1739       Parameters:

CID: 1748     CClass: ConcreteAccessRule     Operation: Link PClass: Interface
PName: 1685     PId: 1685       Parameters:
```

Note: Renumbering will be done for all types of Concretes irrespective of type of operation related to concretes(Create, Link, Unlink, Delete, Affected).

The following example shows the renumbered affected concretes.

```
/>exporttransaction [1751]
Operations:
OID: 589        OClass: DefaultTraffic  OName: Default Traffic Type   Operation:
Unlink       PClass: ClassificationLeaf      PName:  PId: 1740   Parameters:

Concrete Operations:
CID: 1741       Operation: Affected     PClass:         PName:  PId:
CID: 1742       Operation: Affected     PClass:         PName:  PId:

 />exporttransaction [1751]
Operations:
OID: 589        OClass: DefaultTraffic  OName: Default Traffic Type   Operation:
Unlink       PClass: ClassificationLeaf      PName:  PId: 1740   Parameters:

Concrete Operations:
CID: 1748       Operation: Affected     PClass:         PName:  PId:
CID: 1749       Operation: Affected     PClass:         PName:  PId:
```

In the example above affected concretes CIDs 1741 and 1742 are numbered to 1748 and 1749.

In the following example, the transaction that is the object of “exporttransaction” was disabling a concrete that resulted in faults being generated.

Operation:

```
exporttransaction [17414]
Concrete Operations:
CID: 17269     CClass: ConcreteGenericRule    Operation: Modify      PClass:
PName:  PId:    Parameters: Enabled=0

Fault Operations:
FID: 17413      Operation: Create       PClass:         PName:  PId:
Parameters: DefaultText=This user-created interface has had its creation policy
concrete disabled - the interface may not exist on the device, MajorCode=100,
MinorCode=2346, Parameters=, Severity=2, TimeStamp=Thu Jan 31 13:55:45 2008
FID: 17413      Operation: Link PClass: System  PName: 2        PId: 2
Parameters:
```

```
FID: 17413     Operation: Link PClass: SubInterface    PName: 17267    PId: 17267
Parameters:
```

The ListTrans Command

The **listTrans** command lists all the commands entered in the current transaction.

Note that if you run **listTrans** command just after committing a set of commands, **listTrans** returns nothing.

Syntax

The syntax for the **listTrans** command is:

```
listTrans
```

Path

[no default]

Return

The listTrans command returns the list of commands in the format:

```
command object-id [object-id] parameters
```

For example:

```
create /policy:"policy"/domain:"d"/customer:"YDEurope"
```

Errors

[no default]

The Merge Command

The **merge** command merges the content of an existing **TransactionEntry** object to the current local object model. For queued (Pending) transactions, this is required before the transaction can be committed.

During the merge, IP Service Activator tests the validity of the transaction against the local object model. If there is a conflict, IP Service Activator abandons the merge and reports an error. This occurs, for example, if you attempt to merge a transaction that creates an object that already exists.

Merge is a queued command, that is, it is not executed immediately like the other Transaction Module commands.

Syntax

The syntax for the **merge** command is:

```
merge [object-path]
```

For example, to merge the transaction MyQueuedChanges with the local object model:

```
merge /System:"System"/TransactionEntry:"MyQueuedChanges"
```

Path

The path of the **TransactionEntry** object to be merged. If no path is given, the current object is assumed.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified in the path is not a TransactionEntry object
- The transaction in the TransactionEntry object could not be merged with the local object model.

The Rollback Command

The **rollback** command rolls back the changes specified in a **TransactionEntry** object from the current object model.

When you roll back a transaction, its changes are removed from the object model and, where configuration has been installed on network devices, the configuration is removed.

The ability to roll back a committed transaction depends on whether transactions that were subsequently committed are dependent on that transaction's changes. For example, a transaction that created a VPN to which sites and interfaces have subsequently been linked cannot be rolled back. It is not necessary to have merged the content of the TransactionEntry object first.

Rollback is a queued command, that is, it is not executed immediately like the other Transaction Module commands.

Syntax

The syntax for the **rollback** command is:

```
rollback [object-path]
```

For example:

```
rollback /System:"System"/TransactionEntry:"MyQueuedChanges"
```

Path

The path of the TransactionEntry object to be rolled back. If no path is given, the current object is assumed.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified in the path is not a **TransactionEntry** object
- The transaction stored in the **TransactionEntry** object could not be rolled back in the local object model.

The Schedule command

The **schedule** command changes a queued (Pending) **TransactionEntry** object into a scheduled one, or changes the scheduled time for a scheduled **TransactionEntry** object.

Schedule is a queued command; that is, it is not executed immediately like the other Transaction Module commands.

Table 3–9 lists the attributes for the schedule command.

Table 3–9 schedule Command Attributes

Attribute	Type	Description
schedule	DateTime	The date and time when the transaction is to be applied (YYYY/MM/DD HH:MM:00) This parameter must be given in GMT. If the time stated is in the future in local time, but in the past in GMT, the command is executed immediately. Must be in quotes.

Syntax

The syntax for the **schedule** command is:

```
schedule [object-path] schedule=schedule-date
```

For example:

```
schedule TransactionEntry:"MyQueuedChanges" schedule="2001/6/15 20:00:00"
```

Path

The path to the TransactionEntry object to be scheduled. If no path is given, the current object is assumed.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified in the path is not a **TransactionEntry** object
- The TransactionEntry object's state is not Pending or Scheduled.

The Declarative Module

The Declarative Module provides commands that operate on and potentially change objects within the policy and topology trees.

Table 3–10 summarizes the commands that are available.

Table 3–10 Declarative Module Commands

Command	Summary
PreserveMissingInterfaces	Restores to **found** state, the interfaces which are not found.
ApplyRoleAssignmentRules	Applies roles to interfaces and devices in the Network object specified.
copy	Copies the specified object.
create	Creates and names an object.
delete	Deletes the specified object.
link	Links a child object to a parent object.

Table 3–10 (Cont.) Declarative Module Commands

Command	Summary
unlink	Unlinks a child object from a parent object.
modify	Modifies the attributes associated with a specified object.
subscribe/unsubscribe	Sets the system to ignore certain types of objects during the lifetime of a session.
use	Creates an instance of a global template to be used by the specified object.
unuse	Removes the instance of a global template applied on the specified object.

The PreserveMissingInterfaces Command

The **PreserveMissingInterfaces** command restores the interfaces which are 'not found' to a 'found' state. Neither the GUI nor the OSS Integration Manager Interface allow you to manage a device with 'not found' interfaces. These have to either be discovered to be found or must be deleted individually. For more information on preserving missing interfaces, see the IP Service Activator online Help.

Syntax

The syntax for the **PreserveMissingInterfaces** command is:

```
preservemissinginterfaces object_id
```

where *object_id* is the identifier for the device, interface, or sub-interface.

For example:

```
Policy:"Policy"/Domain:"Oracle_Corporation"/Network:"Oracle_Corporation">ls
[3773] ParameterSetInstance:""
[3701] Device:"rot7507-4"
/Policy:"Policy"/Domain:"Oracle_Corporation"/Network:"Oracle_
Corporation">preservemissinginterfaces [3701]
```

Path

The path should be mentioned with device object_id separated by a space.

Return

[no default]

Errors

CommandExecutionError.ERR_CommitFailure: Commit failed on command 1 preservemissinginterfaces

CommandExecutionError.ERR_IncorrectObjectForCommand: The command PreserveMissingInterfaces may not be applied to a Network

The ApplyRoleAssignmentRules Command

The **ApplyRoleAssignmentRules** command applies role assignment rules to devices and interfaces in the current network.

Syntax

The syntax for the **ApplyRoleAssignmentRules** command is:

```
ApplyRoleAssignmentRules
```

For example:

```
cd /Policy:"Policy"/Domain:"MyDomain"/Network:"MyDomain"
ApplyRoleAssignmentRules
commit
```

Path

None, as the command is applied to the current path.

Return

[no default]

Errors

If applied at objects other than a Network object, you will receive the following error or a similar error:

```
CommandExecutionError.ERR_IncorrectObjectForCommand: The command
ApplyRoleAssignmentRules may not be applied to a Domain.
```

The Copy Command

The copy **command** creates a copy of the object and of its direct children, and links the newly-created object to a specified parent.

> **Note:** This command can only be used with policy rules, that is, RuleClassification, RuleAccess and RulePolicing objects.

Syntax

The syntax for the **copy** command is:

```
copy source-object-path dest-object-path
```

where *source-object-path* is the path to the source object and *dest-object-path* is the path to the desired destination.

For example, to copy a rule from one point in the topology to another:

```
copy domain:"europe"/Customer/RuleClassification:"CustomerRule"
domain:"europe"/Network:"England"/RuleClassification:"NetRule"
```

Path

Paths for both source and destination objects must be provided. They must be separated by a space.

Return

[no default]

Errors

[no default]

The Create Command

The **create** command creates and names a new object. If attributes can be defined they override the default attribute settings.

The object is linked to the parent object indicated in the path statement.

Syntax

The syntax for the **create** command is:

```
create object-path [attributes]
```

where *attributes* is an optional list of attribute=value pairs which override the default attributes or supply required attributes.

For example, to create a new customer called Acme Telecom in the Europe domain:

```
create Domain:"europe"/Customer:"Acme Telecom" Remarks="Important Customer"
```

Note: You can set the attributes of the object using either the **create** command or the **modify** command. For example:

```
create Customer:foo Remarks="A Customer" Reference="A Reference"
```

or:

```
create Customer:foo
Modify Customer:foo Remarks="A Customer" Reference="A Reference"
```

However, attributes with an access type of CreateOnly can only be set at the time the object is created. For example, if creating a virtual device:

```
create Device:"NLV2000" IsVirtual=False DeviceType:"Cisco2000"
```

IsVirtual is mandatory, **DeviceType** is optional. The IsVirtual attribute does not appear in the object and cannot be set later.

Path

The path to the new object, ending with the type and name of the object to be created.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The user does not have permission to perform this command
- The object cannot be created at this position

The Delete Command

The **delete** command removes a specified object. All children of the object are unlinked, and any orphaned children are deleted.

Syntax

The syntax for the **delete** command is:

```
delete object-path
```

For example:

- To delete a customer called Acme Telecom in the Europe domain:

```
delete Domain:"europe"/Customer:"Acme Telecom"
```

- To delete the object with an ID of 400:

```
delete [400]
```

Path

The path of the object to be deleted or its ID.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The user does not have permission to perform this command.

The Link Command

The **link** command links two objects in a parent/child relationship. The parent and child objects to be linked are specified in the path.

Syntax

The syntax for the **link** command is:

```
link parent-object-path child-object-path
```

where *parent-object-path* is the path or object ID for the object becoming the parent and *child-object-path* is the path or object ID for the object becoming the child.

For example:

- To link a child object with ID 475 to a parent object with ID 600:

```
link [600] [475]
```

- To link the site “Paris” to the customer “Acme Telecom”:

```
link customer:"Acme Telecom" site:"Paris"
```

Path

Paths for both parent and child objects must be provided. They must be separated by a space.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object types indicated cannot be linked
- The user does not have permission to perform this command

- The child object is already a child of the parent

The Unlink Command

The **unlink** command unlinks a specified child object from its parent object.

Syntax

The syntax for the unlink command is:

```
unlink parent-object-path child-object-path
```

where *parent-object-path* is the path or object ID for the parent object being unlinked and *child-object-path* is the path or object ID for the child object being unlinked.

For example, to unlink a child object with ID 400 from a parent object with ID 300:

```
unlink [300] [400]
```

Path

Paths for both parent and child objects must be provided. They must be separated by a space.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object types indicated cannot be unlinked
- The two objects indicated are not linked

The Modify Command

The **modify** command modifies the attributes of the specified object.

Syntax

The syntax for the **modify** command is:

```
modify object-path parameters
```

where *parameters* is a list of attribute=value pairs which overrides the default attributes.

Attributes may be listed in any order.

For example, to change the comment and name attributes of an object with ID 200.

```
modify [200] comment="this is a comment" name=NewName
```

> **Note:** There is no overflow checking for attributes that are of type U32. If an entered value is negative or too large, then the attribute is changed to the overflowed value.

Path

Path of the object to be modified or its object ID.

Return

[no default]

Errors

An error in one attribute results in all attributes being rejected.

The **CommandExecutionError** may be returned for the following reasons:

- The object or attributes are read only
- The named attribute does not exist on the object

The Subscribe/Unsubscribe Commands

The **subscribe**/**unsubscribe** commands can be used to ignore certain types of objects during the lifetime of a session. For example:

- They can be used when browsing through the EOM using the CLI, to reduce the amount of information displayed on screen. For example, policy elements or topology elements can be hidden.
- They can be used if you want to run **xmlExport** for specific objects only.
- They can be used when waiting for a transaction to happen (**events block=true**), but you want to report on certain objects only, such as ConcreteObjects.
- They can be used to preserve backward compatibility. Certain types can be shown or hidden that were not available in previous versions of OIM.

> **Note:** **Subscribe**/**unsubscribe** are queued commands and must always be committed.

Table 3–11 lists the attributes for the subscribe/unsubscribe command.

Table 3–11 Subscribe/Unsubscribe Command Attributes

Attribute	Type	Description
type	Enum	Set to **all** to specify all objects. Specify an object type to subscribe or unsubscribe from a specific object type.

Syntax

The syntax for the **subscribe**/**unsubscribe** commands is:

```
subscribe type={object-type|all}
unsubscribe type={object-type|all}
commit
```

Path

[no default]

Return

[no default]

Errors

A common query is to wait for policy elements to be applied in the policy server. For example, if you commit a transaction that creates an access rule, you may want to be notified only when the concrete rule objects are created:

```
create RuleAccess:myRule
unsubscribe type=all
subscribe type=ConcreteObject
subscribe type=System
commit
events block=true
```

This makes sure that only events concerning concrete objects are reported. For more information on events command, see "The Events Command".

The Use Command

The **use** command indicates the use of a policy element (such as a PHB group) by a policy target (such as a device). This is achieved by creating an intermediary Instance object.

For example, a target can be configured to use a PHB group by creating a PHBGroupInstance object, which is the child of both the target object and the PHBGroup object.

At present, this command is used only for associating PHB groups and ParameterSetInstance objects with target objects.

Attributes are applied to the Instance object. See "The GetParameters Command".

Syntax

The syntax for the **use** command is:

```
use target-object-path policy-element-path [attributes]
```

For example, to specify that the PHB Group “WRR” is applied to the XYZ network:

```
use network:"XYZNetwork" PHBGroup:"WRR"
```

This creates a PHBGroupInstance object, child of both the XYZNetwork network and the WRR PHB group.

Path

The *target-object-path* is the path of the policy target object that is to use the policy element and *policy-element-path* is the path of the policy element object to be used.

Both paths must be specified separated by a space.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- One of the object types indicated cannot be a target, or cannot be applied

The Unuse Command

The **unuse** command removes the link between a policy element by a policy target by destroying the Instance object created by the **use** command.

Syntax

The syntax for the **use** command is:

```
unuse target-object-path policy-element-path
```

For example, to specify that the PHB Group "WRR" is no longer applied to the XYZ network:

```
unuse network:"XYZNetwork" PHBGroup:"WRR"
```

Path

The *target-object-path* is the path of the policy target object that is using the policy element and *policy-element-path* is the path of the policy element object used.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object type indicated cannot be unused

The Management Module

The Management Module provides commands for discovering and managing devices and subscribing to events.

Table 3–12 summarizes the commands that are available.

Table 3–12 Management Module Commands

Command	Summary
discover	Starts a device discovery operation.
manage	Changes the state of an object to Managed.
unmanage	Changes the state of an object to Unmanaged.
resetCapability	Resets the discovered capabilities for Device and Interface objects.

The Discover Command

The **discover** command begins a discovery operation. The **discover** command can be used to discover specific devices in the physical network and create new device objects to represent them, or to get the attributes of an existing device object from the physical device.

The discover command can be used in one of two ways:

- An IP address and subnet mask or a DNS name can be given and the corresponding device(s) are found and linked to the network object specified in the path.
- The path of an existing object can be given, and the object is rediscovered and its attributes updated if necessary.

The **discover** command is executed immediately rather than being queued to a transaction.

The **discover** command can be applied only to Devices and Networks (as modelled by IP Service Activator).

No attributes are required for *parameters* for a rediscovery. For a new discovery, where the path is an existing network, attributes are required.

Table 3–13 lists and describes the possible attributes for the discover command.

For more on the **discover** command, see "Discovering Devices", and for more information on the discovery process, see *IP Service Activator User's Guide*.

Table 3–13 Discover Command Attributes

Attribute	Type	Description
Type	Enum	The type of discovery to perform. Can take the following values: ■ 0 = Refresh: performs a rediscover of all discoverable objects in the domain ■ 1 = Discover: performs a "normal" SNMP discovery (this is the default). ■ 3 = LocalSegment: performs an SNMP discovery of the routers on the local segment. **Note:** an SNMP agent must be running on the server machine ■ 4 = GetCapabilities: fetches the capabilities of the discoverable objects. **Note:** Does not rediscover capabilities where they have already been fetched. **Note:** GetCapabilities will only return results when applied against Network objects, not individual Devices. ■ 5 = Stop: stops any current discovery process
DnsName	String	DNS name of the object to be discovered. Either this or IpAddress must be supplied when Type=Discover.
IpAddress	IpAddress	IP address of the object or subnet to be discovered. Either this or DnsName must be supplied when Type=Discover.
Mask	U32	The subnet mask of the object to be discovered. Defaults to 32 if the IP address is a device, 24 if it is a subnet.
SnmpProfileName	String	The SNMP Profile to use for discovery. Note: An error message is returned if you specify any of these parameters: ReadCommunity, Retries, Timeout, MaxRepetitions or SnmpVn, in the same command statement as SnmpProfileName.
ReadCommunity	String	The SNMP Read community of the object(s) to be discovered. Default value is "public".
Retries	U32	The number of retries performed for each object. Default value is 2.
Timeout	U32	The time (in seconds) before a discovery attempt times out. Default value is 3.
SnmpVersion	Enum	SNMP version: ■ V1V2c = both SNMP versions (default) ■ V1 = Version 1 ■ V2 = Version 2c

Table 3–13 (Cont.) Discover Command Attributes

Attribute	Type	Description
AccessStyle	String	The means of accessing and authenticating with the object - required when fetching the capabilities of each object. Valid values are: ■ None: capabilities are not fetched ■ NamedUser ■ Anonymous ■ TACACS ■ SNMPv1 ■ SNMPv2c ■ SSH ■ RSA_SSH ■ passwordOnly If no AccessStyle parameter is set, then the discover is performed with the settings that were set previously.
InheritsSecurity	Boolean	If True, security settings are inherited from the network object. If False, security settings must be set for the command.
WriteCommunity	String	The SNMP Write community to use when fetching capabilities. Required if AccessStyle set to NamedUser, SNMPv1, or SNMPv2c.
UserName	String	The user name for login when fetching capabilities. Required if AccessStyle set to NamedUser or SSH.
LoginPassword	String	The Login password to use when fetching capabilities. Required if AccessStyle set to NamedUser, Anonymous, SSH or passwordOnly.
EnablePassword	String	The Enable password to use when fetching capabilities. Required if AccessStyle set to NamedUser, Anonymous, or SSH.
Filename	String	Path of topology import file. Not used; file import is not currently supported.
RsaPrivateKey	String	Name of private key file. Required only when AccessStyle is RSA_SSH.
MaxRepetitions	U32	Specifies the maximum number of rows that will fetch from a network resource in a single request when SNMP V2c is used for discovery. Permitted range for this filed is 1 to 100. Decrease the value of Max-Repetitions if the routers don't respond to the default value of 100.

Syntax

The syntax for the discover command is:

```
discover object-path [parameters]
```

For example:

- To discover a device as a child object of the root network, assuming the root network ID has an object ID of 250:

  ```
  discover [250] ipaddress=10.0.0.30
  ```

- To discover all devices on a subnet:

  ```
  discover [250] ipaddress=10.0.0.0
  ```

- To discover a device and get the capabilities, using anonymous login:

  ```
  discover [250] ipaddress=10.0.0.30 accessstyle=anonymous loginpassword=apasswd
  enable password=anotherpassword
  ```

- To discover a device, inheriting security settings from the network object:

```
discover policy:"Policy"/domain:"DDTest"/network:"DDTest"
ipaddress=192.168.27.1 accessstyle="TACACS" inheritssecurity=True
```

- To rediscover an existing device with the object ID of 500:

```
discover [500]
```

- To discover a device using an SNMP Profile:

```
discover ipaddress=192.168.27.1 SnmpProfileName=SNMP_profile3
```

Path

The *object-path* can be one of two things. For a new discovery, *object-path* is the path or ID of the object that the already-discovered objects are to be linked to (must be a network object). For a rediscovery, *object-path* is the path or ID of the object to be rediscovered.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- Incorrect combination of attributes.
- Path specified is not a discoverable object or an object a to which a discoverable object can be linked.

The Manage Command

The **manage** command changes the state of the specified object to Managed. This command can only be applied to a Device object.

Syntax

The syntax for the **manage** command is:

```
manage [object-path]
```

For example, to manage the device with an object ID of 3675:

```
manage [3675]
```

Path

The *object-path* is the path or object ID of the object to be managed.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The specified object is already managed.
- The object cannot be managed.

The Unmanage Command

The **unmanage** command changes the state of the specified object to Unmanaged. This command can only be applied to a Device object.

Syntax

The syntax for the **unmanage** command is:

```
unmanage [object-path]
```

For example, to unmanage the current object:

```
unmanage
```

Path

The *object-path* is the path or ID of the object to be unmanaged.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The specified object is already unmanaged.
- The object cannot be managed.

The ResetCapability command

The **resetCapability** command resets the capabilities discovered for a specific target. This command can only be applied to a Device or Interface object. Ensure that the device is in the unmanaged state before applying the resetCapability command.

After the command is applied, the capability of the target device and its child interfaces or the target Interface are linked to a default (NULL) capability.

Syntax

The syntax for the **resetCapability** command is:

```
resetCapability [object-path]
```

Path

The *object-path* is the path or ID of the Device or Interface object to be unmanaged.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The specified object is invalid for this command.

The Navigation Module

The Navigation Module provides commands that operate on objects derived from both the policy and topology trees. Navigation commands do not alter any objects. The commands are executed directly rather than being queued in a transaction.

Table 3–14 summarizes the commands that are available.

Table 3–14 Navigation Module Commands

Command	Summary
find	Locates objects in the EOM.
findParameters	Finds objects of a particular type that have parameters with certain values.
getAttributes	Lists the attributes of an object.
getChildren	Lists the children of an object.
getID	Returns the ID of an object.
getName	Returns the name of an object.
getParameters	Returns a list of parameters that are applied to the target object.
getParents	Lists the parents of an object.
getPath	Returns the path to the current position in the EOM.
setPath	Sets the current path within the EOM.
getTargets	Returns the targets to which a ParameterSetInstance applies.
xmlExport	Retrieves a sub-tree of the EOM and outputs in XML format.

The Find Command

The **find** command locates an object in the object model and returns its path. The match is made against the object name and/or object attributes.

The search string may contain wildcard tokens, where an asterisk represents any number of characters and a question mark represents a single character.

The search starts at the point specified by the object path and can search up or down the topology tree.

You can use the Depth and Limit attributes to refine the scope of a search.

For more examples of using the find command, see "Finding Objects".

Table 3–15 lists the possible attributes.

Table 3–15 Find Command Attributes

Attribute	Type	Description
FindDirection	Enum	Indicates the direction of the search. Possible values are: Parent: to search upwards through the hierarchy. Child: to search downwards through the hierarchy (this is the default).
Various	Various	Attribute=value pairs on which to search. Wildcards are permitted in string arguments. This attribute is optional if an object is specified.
Depth	Integer	Limits the number of levels in the object tree hierarchy that are included in a search. For example, if you set Depth to 1, the search includes only immediate children. If Depth is not specified, a parallel search is executed across and down all siblings.

Table 3–15 (Cont.) Find Command Attributes

Attribute	Type	Description
Limit	Integer	Limits the number of returned objects in order to improve the performance of the Find command. For example, setting the Limit attribute to 1 finds and returns only the first object.

Syntax

The syntax for the **find** command is:

```
find [object-path] search-string [attributes]
```

Values for *attributes* may be specified to narrow the search. Only objects matching the search string and any specified attributes are returned.

For example:

- To search for interfaces with the name Serial0:

  ```
  find . interface:"Serial0"
  ```

- To search up the tree from the current location for any customer objects:

  ```
  find . Customer:"*" finddirection=parent
  ```

- To search the domain called Acme to find VPN objects starting with the string "Mari" and with an IP address starting with 212:

  ```
  find domain:"Acme" "vpn:Mari*" direction=child ipAddress="212*" vpnType=Mesh
  ```

  returns:

  ```
  [276]   vpn:"Marigold"
  [923]   vpn:"Mario"
  ```

Path

The *object-path* is the path or ID of an object. If no path is given, the current object is assumed.

Return

The find command returns a list of all objects that meet the search criteria.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The MatchConditions are malformed.
- The find command could not return anything. For example, an Object Type used in the command does not exist.

The FindParameters command

The **findParameters** command locates an object in the object model that has parameters set to particular values and returns its path. The match is made against the object name and/or object parameters.

Parameters are applied to an object when a ParameterSetInstance object is linked to it or to a parent object from which it inherits.

The search string may contain wildcard tokens, where an asterisk represents any number of characters and a question mark represents a single character.

The search starts at the point specified by the object path and can search up or down the topology tree.

When the **findParameters** command is run without parameters, it has the same effect as the **find** command.

Table 3–16 lists the parameters that can be specified along with their corresponding attributes.

Table 3–16 FindParameters Command Parameters and Attributes

Attribute	Type	Description
Direction	Enum	Indicates the direction of the search. Possible values are: Parent: to search upwards through the hierarchy. Child: to search downwards through the hierarchy (this is the default).
Various	Various	Attribute=Value pairs on which to search. Wildcards are permitted in string arguments.

Syntax

The syntax for the **findParameters** command is:

```
findParameters [object-path] search-string [parameters]
```

Values for *parameters* can be specified to narrow the search. Only objects matching the search string and any specified parameters are returned.

For example, to find all interfaces with the parameter OCH_MeasureCBQos=True:

```
findParameters / Interface:"*" OCH_MeasureCBQos=True
```

Path

The *object-path* is the path or ID of an object. If no path is given, the current object is assumed.

Return

The findParameters command returns a list of all the objects that meet the search criteria.

Errors

[no default]

FindParameters Command Errors

The **CommandExecutionError** may be returned for the following reasons:

- The MatchConditions are malformed
- The **findParameters** command could not return anything. For example, an object type used in the command does not exist

The GetAttributes command

The **getAttributes** command lists the attributes for a specified object.

Syntax

The syntax for the **getAttributes** command is:

```
getAttributes [object-path]
```

For example, the command to return the attributes for a selected EventSubscription object:

```
getAttributes /System:"System"/EventSubscription:"mySubscription"
```

returns:

```
    rcw String     Name               = "mySubscription"
    rcw String     Description        = ""
    rcw Boolean    Enable             = True
    rcw Boolean    SendPendingEvents  = False
    rcw Enum       DeliveryType       = Netcool
    rcw String     DeliveryDetails    = ""
    rcw U32        Id                 = 2
```

Path

The *object-path* is the path or ID of an object for which the list of attributes is required. If no path is given, the current object is assumed.

Return

The **getAttributes** command returns a list of attributes for the specified object, including their data type and access rights (r=read, c=create, w=write).

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The GetChildren command

The **getChildren** command lists all child objects that are linked to the specified object.

Syntax

The syntax for the **getChildren** command is:

```
getchildren [object-path]
```

For example, to return the children of the current object:

```
getChildren
```

returns:

```
[697]   domain:"zeus"
[745]   domain:"diana"
[211]   domain:"Aphrodite"
```

Path

The *object-path* is the path or ID of an object for which the list of children is required. If no path is given, the current object is assumed.

Return

The **getChildren** command returns a list of children of the specified object, returned as a sequence of objects.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found

The GetID command

The **getID** command returns the ID of an object.

Syntax

The syntax for the **getID** command is:

```
getid [object-path]
```

For example, to return the ID number of the VPN called Executive:

```
getID /domain:"europe"/customer:"myCustomer"/vpn:"executive"
```

returns:

```
[8359]
```

Path

The *object-object* path is the path to the object for which the ID is needed. If no path is given, the current object is assumed.

Return

The **getID** command returns the requested object ID number.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found

The GetName Command

The **getName** command returns the object name for the specified object.

Syntax

The syntax for the **getName** command is:

```
getName [object-path]
```

For example:

- To return the name of the parent object:

  ```
  getName ..
  ```

- To return the name of the object with an ID of 8359:

  ```
  getName [8359]
  ```

  The name is returned in the following format:

```
vpn:"executive"
```

Path

The *object-path* is the path or ID of the object for which the name is required. If no path is given, the current object is assumed.

Return

The name of the current object.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The GetParameters Command

The **getParameters** command retrieves a list of parameters that are applied to the target object.

Parameters are applied when "ParameterSetInstance" objects are linked to the object, or to an object higher in the hierarchy. The Levels attribute of the ParameterSetInstance object determines which type of object the parameters are applied to.

Before using the **getParameters** command, a parameter set must be applied using the **use** command. Specific values of parameters may be set using the **use** command, or by modifying the ParameterSetInstance object created. See "The Use Command" for further information.

Syntax

The syntax for the **getParameters** command is:

```
getParameters [object-path]
```

For example, to apply the parameter set object (ID=15) to a network object (ID=350):

```
use [350] [15] name="Instance" OCH_NetflowEnabled=True Levels=4
```

This creates a ParameterSetInstance object with the name "**Instance**" which is a child of both the Network object specified and the ParameterSet. Only the **OCH_NetflowEnabled** and **Levels** attributes are set; other attributes are set to the default value. **Levels=4** means that the parameters are applied at the interface level within the specified network.

The ParameterSetInstance object created must then be linked to a DeviceRole and an InterfaceRole. For example, assuming the newly created ParameterSetInstance object has an ID of 555:

```
link [555] [23]
link [555] [28]
```

Using **getParameters** on one of the interfaces with the correct roles shows the applied parameters:

```
getParameters [2456]
```

Could return the following:

```
r-- Boolean     OCH_MeasureCBQoS     = False
```

```
r-- Boolean     OCH_MeasureCarQoSMIB = False
r-- Boolean     OCH_MeasureMIB2Stats = False
r-- Boolean     OCH_MeasureNetflowEnabled = False
r-- U32     OCH_NetflowVersion = 1
r-- U32     OCH_NetflowAggregation = 0
r-- U32     OCH_NetflowCacheSize = 0
r-- U32     OCH_NetflowTimeoutActive = 0
r-- U32     OCH_NetflowTimeoutInactive = 0
```

Path

The *object-path* is the path or ID of the object for which the list or parameters is required.

Return

Returns the parameters that are applied to the specified target object.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The GetParents command

The **getParents** command lists all the parents of the specified object.

Syntax

The syntax for the **getParents** command is:

```
getparents [object-path]
```

For example, to list the parents of the object with an ID of 8349:

```
getParents [8349]
```

The details are returned in the following format:

```
[653]   domain:"europe"
```

Path

The *object-path* is the path or ID of the object for which the list of parents is required. If no path is given, the current object is assumed.

Return

The **getParents** command returns the list of all parents of the specified objects as a sequence of objects.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The GetPath command

The **getPath** command returns the path to a given object ID or the current position in the object hierarchy.

Table 3–17 lists the possible attribute.

Table 3–17 getPath Command Attributes

Attribute	Type	Description
Unique	Boolean	Returns a unique path with the full ID in the path instead of the object name.

Syntax

The syntax for the **getPath** command is:

```
getPath [object-path]
```

For example, to return the path of the current object:

```
getPath
```

The details are returned in the following format:

```
/domain:"europe"/customer:"myCustomer"/vpn:"executive"
```

Path

The *object-path* is the ID of the object for which the path is required. If no value is given, the current object is assumed.

Return

The **getPath** command returns the current path, or the path to the given object. If an object ID was given that is in the current path or is a child of the current object, then the path returned is constructed from the current path.

If any other ID is given and there is more than one possible path to the object then a path is selected arbitrarily from the set of possible paths.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The SetPath Command

The **setPath** command sets the current path within the object hierarchy.

Syntax

The syntax for the **setPath** command is:

```
setPath [object-path]
```

For example, to set the path to the "Executive" VPN:

```
setPath customer:"myCustomer"/vpn:"executive"
```

returns:

```
/domain:"europe"/customer:"myCustomer"/vpn:"executive"
```

Path

The *object-path* is the path or ID of the object for which the path is to be set. Table Table 3–18 shows the symbols that may be used in the path.

Table 3–18 Symbols Used in the SetPath Command

Symbol	Description
.	Current path
..	Path to parent object
/	Object separator
Object name	Set the path to the location of the specified object
Object ID	Set the path to the location of the object with the specified ID

Return

Returns the current path.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The GetTargets command

The **getTargets** command retrieves the targets that a ParameterSetInstance object applies to. A ParameterSetInstance is applied to a parent object, but the parameters may be applied to other objects.

For example, if you link a ParameterSetInstance to a Network object, it would not be applied to the Network object, but to devices or interfaces that inherit from the Network object. (The usual rules of inheritance apply.) The **getTargets** command returns a list of those objects the ParameterSetInstance applies to.

For more information, see "Applying Parameter Sets".

Syntax

The syntax for the **getTargets** command is:

```
getTargets [object-path]
```

For example, to return a list of the interfaces where the ParameterSetInstance object has been applied:

```
getTargets [2380]
```

returns:

```
[302] Interface: "Ethernet1"
[318] Interface: "Ethernet2"
```

Path

The *object-path* is the path or ID of the ParameterSetInstance object for which the targets are required.

Return

[no default]

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found
- You do not have permission to view the object

The XmlExport command

The xmlExport command outputs the external object model from the target object in XML format. The output is formatted as follows:

```
<ObjectType attribute1="value1" attribute2="value2"...>
<ObjectType attribute1="value1" attribute2="value2"...>
<ObjectType attribute1="value1" attribute2="value2"...>
</ObjectType>
```

If the object has children, they are shown further indented and the **</ObjectType>** closes the definition.

If the object has no children, the tag takes the form **<ObjectType attributes.../>**

Syntax

The syntax for the **xmlExport** command is:

```
xmlExport [object-path] [xmlEscape=boolean-value] [compress=boolean-value]
```

For example, assuming the object with an ID of 400 is a domain:

```
xmlExport [400] xmlEscape=true
```

might return:

```
<Domain Id="400" name"Europe">
<Customer Id="500" Name="Cust1" Remarks="A customer">
<Site Id="600" Name="Site1"/>
</Customer>
</Customer Id="700" Name="Cust2"/>
</Domain>
```

In this example, the domain (ID=400) has two children which are both customers. One customer (ID=500) has a child (a site), but the other customer (ID=700) does not.

Path

The *object-path* is the path or ID of an object. If no path is given, the current object is assumed.

Return

The **xmlExport** command returns details of all objects below the target object in the hierarchy in XML format.

Errors

The **CommandExecutionError** may be returned for the following reasons:

- The object specified is not found

- You do not have permission to view the object

XML Standard Encoding

The xmlEscape parameter governs whether XML standard encoding escape characters will or will not be used. In releases of IP Service Activator prior to 4.0, the same non-standard encoding scheme was used when writing attribute values on the CORBA channel. The output of the **xmlExport** command was not compatible with the XML standard, and could not be processed correctly by off-the-shelf libraries.

Starting with release 4.0 of IP Service Activator, the OIM now follows the XML standard way of encoding certain characters when using the ISO-8859-1 charset, as described in section 2.4 of the XML standard document.

Table 3–19 lists the escape characters used starting with release 4.0.

Table 3–19 XML Standard Escape Characters

Sequence	Character Represented
&amp;	&
&quot;	"
&lt;	<
&gt;	>

To override this new behavior and preserve the pre-4.0 release encoding, use the value **false** for the xmlEscape parameter.

xmlExport Compression

The compress parameter allows you to compress the data stream created and sent to the client by the **xmlExport** command. Using the value **true** for the compress parameter compresses the stream to BASE64 using the standard ZLIB format, with no header. Compression typically reduces the amount of memory used and the size of the data to 10% of its original size resulting in a decrease of execution time for the operation.

The Utility Module

The Utility Module provides commands that assist in the operation of OIM.

Table 3–20 summarizes the commands that are available:

Table 3–20 Utility Module Commands

Command	Summary
alias	Creates aliases for commands.
file	Sends the output to a file instead of giving a screen display.

The Alias command

The **alias** command defines an alias for a particular command. This allows you to use an alternative or a short form of a command if you prefer.

Aliases apply only to the current session; they are removed when the session terminates.

Syntax

The syntax for the **alias** command is:

```
alias [alias=command]
```

where *alias* is the alias to be used and *command* is the existing command for which the alias is to be used. If no parameters are entered, the alias command returns all current aliases.

For example:

- To set up the alias "cd" to be used as an alternative to the **setPath** command:

  ```
  alias cd=setpath
  ```

- To list all current aliases:

  ```
  alias
  ```

Path

[no default]

Return

[no default]

Errors

[no default]

The File Command

The **file** command sends the output to a file instead of giving a screen display.

Syntax

The syntax for the **file** command is:

```
/>find / SystemUser:"*"
     [42] SystemUser:"transactionMonitor"
     [3660] SystemUser:"mcurtis"
     [40] SystemUser:"admin"
     [2710] SystemUser:"testuser1"
     [4092] SystemUser:"steve"
     [2712] SystemUser:"testuser2"

/>file /tmp/SystemUser.txt find / SystemUser:"*"

/>exit
 -bash-3.00$ cat /tmp/SystemUser.txt
     [42] SystemUser:"transactionMonitor"
     [3660] SystemUser:"mcurtis"
     [40] SystemUser:"admin"
     [2710] SystemUser:"testuser1"
     [4092] SystemUser:"steve"
     [2712] SystemUser:"testuser2"
 -bash-3.00$
```

Path

[no default]

Return
[no default]

Errors
[no default]

4

The External Object Model

This chapter provides details of the External Object Model (EOM).

General Concepts

This section provides the general concepts of the EOM.

Purpose of the External Object Model

The EOM is a simplified version of Oracle Communications IP Service Activator's internal object model. It defines all the objects that can be accessed or updated by external applications, including their attributes and the relationships between them. The EOM is a subset of the object model, allowing user programs to create and access data objects without requiring knowledge of the underlying complexity of the entire object model.

Overall Structure

The EOM is divided into three major categories:

- **Policy**: The Policy Model contains objects for defining QoS and security in terms of rules and general QoS mechanisms. It is used in conjunction with the Topology Model to apply QoS and security policies to actual devices in a real network.
- **Topology**: The Topology Model contains objects that represent the network topology of the actual managed network, such as VPNs, devices, interfaces and VC objects.
- **System**: The System Model contains objects that represent the IP Service Activator system components and associated system management objects.

Object Notation

Objects within the EOM are described under the following headings:

- **Object diagram**: showing the relationships between objects.
- **Attributes**: the name, data type, default value, and access type of each attribute of the object, plus any additional explanatory remarks. Note that attribute names must not include spaces, for example, PacketMarkingName. Case is not significant and is used in this document for clarity only.
- **Inheritance**: a definition of the attributes that are inherited from other objects (see "Object Inheritance and Abstract Objects").

Access Types

Each attribute has one of the Access Types listed in Table 4–1.

Table 4–1 Attribute Access Types

Type	Meaning
RO	Read Only.
RW	Read/Write.
RC	Read/Modify on create only.
WO	The attribute is masked but can be edited (for example, passwords).
CO	Can be set on object creation only. The attribute will not appear on the object.

Data Types

Each attribute has one of the Data Types listed in Table 4–2.

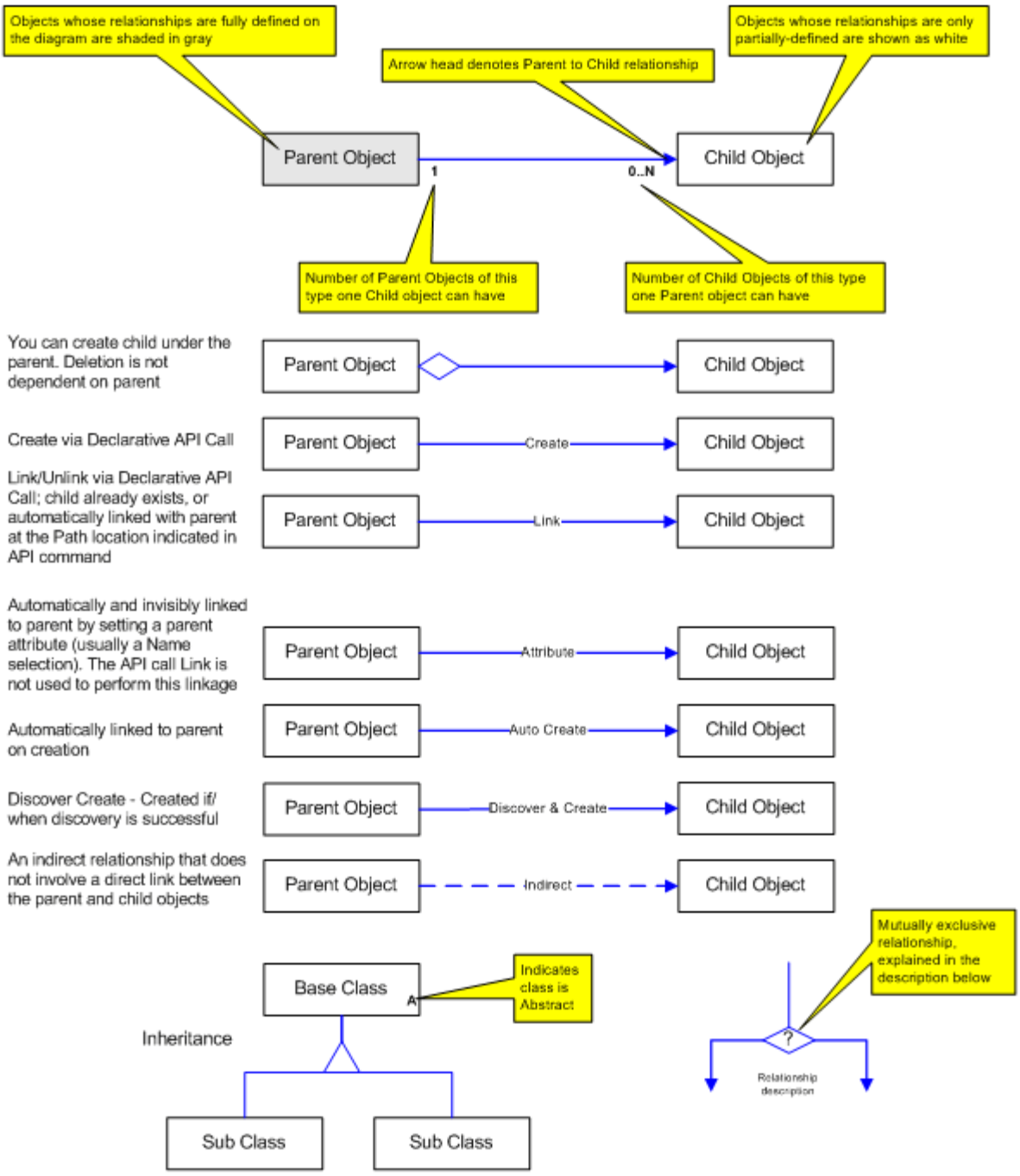
Table 4–2 Attribute Data Types

Type	Meaning
U32	Unsigned 32-bit integer. In some cases the value may be restricted to a smaller range than 32 bits. In these cases the range is noted in the text.
String	A string, containing any alphanumeric characters, generally less than 127 characters. In some cases the maximum length is shorter or longer, and is detailed in the relevant parts of this document.
Enum	A string representing one member from a set of enumerated values.
Boolean	Boolean, can be True or False.
IPAddress	IPv4 address or subnet mask, in the format *nnn.nnn.nnn.nnn* where nnn <=255. IPv6 address is eight groups of four hexadecimal digits (for example, 2001:0db8:85a3:08d3:1319:8a2e:0370:7334)
DateTime	Date and time, in the format YYYY/MM/DD HH:MM:SS. All times in OIM are displayed and set as GMT.

Key to Object Diagrams

Relationships between objects are represented diagrammatically. Figure 4–1shows the standard way in which information is represented.

Figure 4–1 Object Diagram Key



Linking by Attribute

To simplify the manipulation of objects, the EOM has the ability in certain cases to link objects by attribute. This means that an attribute of an EOM object may actually be the name of another child object, and modifying that name will unlink the current child and link a new child with the specified name. This actual link is hidden from the user, and is performed automatically.

If such an attribute is modified then the attribute must be changed to the name of an existing object of the correct type. The system will locate the new object to be linked by searching, usually the domain or the policy object.

In most cases the attribute may be modified to an empty string. The child object will then be unlinked, and no replacement object will be linked.

The descriptions of the EOM object indicate which attributes (if any) automatically link to other objects.

Object Inheritance and Abstract Objects

To simplify the representation of data, a concept of abstract objects is used. Abstract objects hold common attributes that are inherited by all child objects.

Abstract objects cannot themselves be accessed via the OIM command set.

Table 4–3 lists and describes the abstract objects.

Table 4–3 Abstract Objects

Object	Purpose
Object	Inherited by all EOM objects; common attributes (Name and ID).
Traffic	Inherited by all traffic type objects (TrafficGroup, TrafficCompound, TrafficMime, TrafficPort, TrafficURL, TrafficApplication, TrafficSubApplication, TrafficPacketMarking or TrafficDomainName, TrafficVlan, TrafficInputInterface).
Rule	Inherited by all rule objects (RuleAccess, RuleClass, RuleGeneric or RulePolice); specifies common rule attributes except RuleGeneric, which specifies a configuration policy.
VCEndpoint	Inherited by all VC endpoint objects (VCEndpointFr or VCEndpointAtm).
Role	Inherited by RoleDevice and RoleInterface objects.
ClassificationBase	Inherited by Classification and ClassificationGroup objects.
EventFilter	Inherited by EventFilterAttributeChange and EventFilterFaultMask objects.

Object Reference

All objects within the EOM inherit the ID attribute and the Name attribute from the abstract object. Table 4–4 gives details about these attributes.

Table 4–4 Attributes Inherited from Abstract Objects

Attribute Name	Type	Default	Access	Explanation
Id	U32	0	RO	Unique object reference, which remains unique for the lifetime of the object.
Name	String	[no default]	RO/RW*	Name of object.

*Access varies according to object. See details of each object.

Object Inheritance

Each object inherits attributes from its parent object. This is represented using the following notation:

object.parent.[*parent*]

For example:

RuleAccess.Rule.Object

indicates that the RuleAccess object inherits attributes from the Rule object, which in turn inherits attributes from the abstract object.

Summary of Objects

This section provides a summary of the objects.

Policy Model Objects

Table 4–5 describes the Policy Model objects.

Table 4–5 Policy Model Objects

Object	Purpose
Account	Represents a user, host or subnet account or an account group: a source or destination point to which rules can be applied.
Classification	Represents a classification object, the association of a source and destination IP address and a traffic between those two hosts.
ClassificationFolder	Represents a classification folder, used to contain classification objects and other classification folders
ClassificationGroup	Represents a group of classification objects.
ClassificationOrder	Represents a sequence of classification objects.
ConcreteObject	Represents the actual implementation of a policy element or VPN; automatically created when an object is applied to a point in the network.
Cos	Represents a class of service.
COSFolder	Represents a class of service folder, used to contain class of service objects and other class of service folders
Customer	Represents a customer, to which VPNs and sites are linked.
CustomerFolder	Represents a customer folder, used to contain customers and other customer folders.
Domain	Represents a domain, the logical organization for which policies and services can be defined.
DriverScript	Represents a driver script, a set of commands defined in Python, that when applied to a device, results in IOS command script being generated.
DriverScriptFolder	Represents a driver script folder, used to contain driver scripts and other driver script folders.
InterfacePolicyRegistration	Represents the interface policy registration that informs the IP Service Activator core about a generic policy and its abilities.
PacketMarking	Defines different classes of service.

Table 4–5 (Cont.) Policy Model Objects

Object	Purpose
ParameterSet	Represents a set of parameters that can be given different values when applied to different objects.
ParameterSetInstance	A template object defining a set of parameters that can be applied to different objects.
Period	Identifies the time, date and or days of the week to which a rule is to apply.
PHB	Represents the application of a specific queuing mechanism to a class of service
PHBAtm	Represents the queuing mechanism of ATM traffic shaping to an interface.
PHBFrts	Represents the queuing mechanism of a Frame Relay traffic shaping to an interface.
PHBGroup	Represents a PHB group: an implementation of a specific queuing/shaping mechanism available at an interface.
PHBGroupFolder	Represents a PHB Group folder, used to contain PHB Groups and other PHB Group folders.
PHBGroupInstance	Represents a particular application of a PHB group on an object: adds the notion of order between PHB groups.
PHBGroupMqc	Represents an MQC PHB group (the application of a queuing/shaping mechanism via Cisco's Modular QoS CLI.)
PHBMqc	Represents the application of a specific MQC mechanism to a class of service.
PHBPolicingAction	Defines a policing action for use with an MQC PHB group that applies policing.
PHBWred	represents the application of a WRED mechanism to an interface.
Policy	Represents the root object of the policy tree.
PtToPtL2Martini	Represents the point-to-point Layer 2 Martini VPN service
RoleDevice	Represents a given role for a device which defines what policy can be applied to the device.
RoleFolder	Represents a role folder, used to contain interface and device role objects and other role folders
RoleInterface	Represents a given role for an interface which defines what policy can be applied to the device.
RtNumber	Represents a Route Target, as used in MPLS VPNs.
RuleAccess	Represents an access rule, used to deny or permit access to the network for specific identified traffic.
RuleClassification	Represents a classification rule, used to classify, mark, and manage network traffic.
RuleGeneric object	Represents a configuration policy.
Policy Type Object	Holds information regarding the creation of RuleGeneric objects.
Policy Type Folder Object	Categorizes policy types within the Object Model.
RulePolicing	Represents a policing rule, used to police traffic associated with a certain classification.

Table 4–5 (Cont.) Policy Model Objects

Object	Purpose
SAAOperation	Represents the parameters used to configure an SAA operation.
SAATemplate	Represents a parent object of SAAOperation. It groups a number of SAAOperation objects.
Service Group	Represents an Ethernet Virtual Connection (EVC).
Site	Represents a site: a physical location defined by one or more devices and interfaces.
SiteFolder	Represents a folder used to contain site objects and site subfolders.
SiteHub	Represents the hub role of a site in a VPN.
SiteL2	Represents a Layer 2 Site.
StaticRoute	Represents a static route defined for a VPN site.
Tls	Represents a Transparent LAN Service (Layer 2 VPN). The Tls object is linked to a Customer, and SiteL2 objects are linked to it.
TrafficApplication	Represents an application-based traffic type.
TrafficCompound	Represents a compound traffic type: a traffic type that is a combination of two or more traffic types.
TrafficDomainName	Represents a domain-based traffic type.
TrafficGroup	Represents a traffic type group: a logical organization of traffic types into a folder-like structure.
TrafficMime	Represents a MIME-based traffic type.
TrafficPacketMarking	Represents a traffic type based on packet marking (DiffServ codepoint, IP Precedence or MPLS Experimental marking).
TrafficPort	Represents a port-based traffic type.
TrafficSubApplication	Represents a subapplication-based traffic type.
TrafficURL	Represents a URL-based traffic type.
TrafficVlan	Represents traffic based on the Layer 2 VLAN identification number.
TrafficInputInterface	Represents traffic based on the input interface or range of interface through which the traffic or IP packet enters.
Vpn	Represents a virtual private network, defined by a set of interfaces and/or CE routers.

Topology Model Objects

Table 4–6 describes the Topology Model objects.

Table 4–6 Topology Model Objects

Object	Purpose
BgpAggregateAddress	Represents network statements that BGP will advertise for the site.
Device	Represents a device within the network: a network node that forwards IP packets, that is, a router or Layer 3 switch.
DeviceCapabilities	Represents the capabilities of a device.
DeviceType	Represents the model of a device, for example, Cisco 2500.

Table 4–6 (Cont.) Topology Model Objects

Object	Purpose
EigrpRedistribution	Redistribution attributes (delay, reliability, loading and mtu) from other protocols (connected, static, Bgp, Rip) into Eigrp.
EthernetVlan	Represents an Ethernet VLAN.
Interface	Represents an interface on a device.
InterfaceCapabilities	Represents the capabilities and characteristics of an interface on the device.
SAP	Service Application Point. Provides a location to which to attach a VRF in an interface-less site.
Network	Represents a network, a logical object within a domain comprising a number of devices and, optionally, sub-networks.
OspfSummaryAddress	Represents the advertising of OSPF routes for redistribution as a summary address.
Segment	Represents the locally-connected network segment on an interface.
SubLayer	Represents one protocol sublayer of an interface.
SubInterface	Represents a sub-interface on an interface.
CreationMarkerSubInt	Represents a sub-interface with a PVC.
CreationMarkerVcFr	Represents an interface with a Frame-Relay VC endpoint.
Topology	Represents the root object of the Topology tree.
VcEndpointAtm	Represents an ATM PVC endpoint.
VcEndpointFr	Represents a Frame Relay PVC endpoint.
VlanInterface	Represents a VLAN interface.

System Model Objects

Table 4–7 describes the System Model objects.

Table 4–7 System Model Objects

Object	Purpose
Component	Represents an IP Service Activator component: Component Manager, Policy Server, Proxy Agent, Device Driver, Event Handler, Integration Manager, System Logger, or Measurement Component.
EventCollector	Represents a monitoring place in the External Object Model; defines the objects on which faults are to be monitored.
EventFilterAttributeChange	Represents a specific filter associated with an event collector; defines a particular attribute to monitor.
EventFilterFaultMask	Represents a specific filter associated with an event collector; defines a specific fault or a type of fault to monitor.
EventSubscription	Represents an event subscription, defining the way in which an external user subscribes to fault and event reporting.
ExternalSystem	Represents an external system or component.
Fault	Represents a fault that has been reported from an IP Service Activator component.

Table 4–7 (Cont.) System Model Objects

Object	Purpose
Options	Represents system-wide options.
Root	Represents the top of the tree of objects.
SnmpProfile	Represents a user-defined profile of SNMP attributes used to discover a group of devices or individual devices.
System	Represents the root object of the System tree.
SystemUser	The system user object is used to create new users and set security restrictions.
SystemUserGroup	A system user group defines the access level that its members have within IP Service Activator.
TransactionEntry	Represents a queued or scheduled transaction.

The Policy Model

This section describes the Policy Model objects.

Policy Object

The Policy object represents the root of the entire policy tree.

The Policy object has the following object inheritance:

```
Policy.Object
```

Figure 4–2 shows the Policy object diagram.

Figure 4–2 The Policy Object Diagram

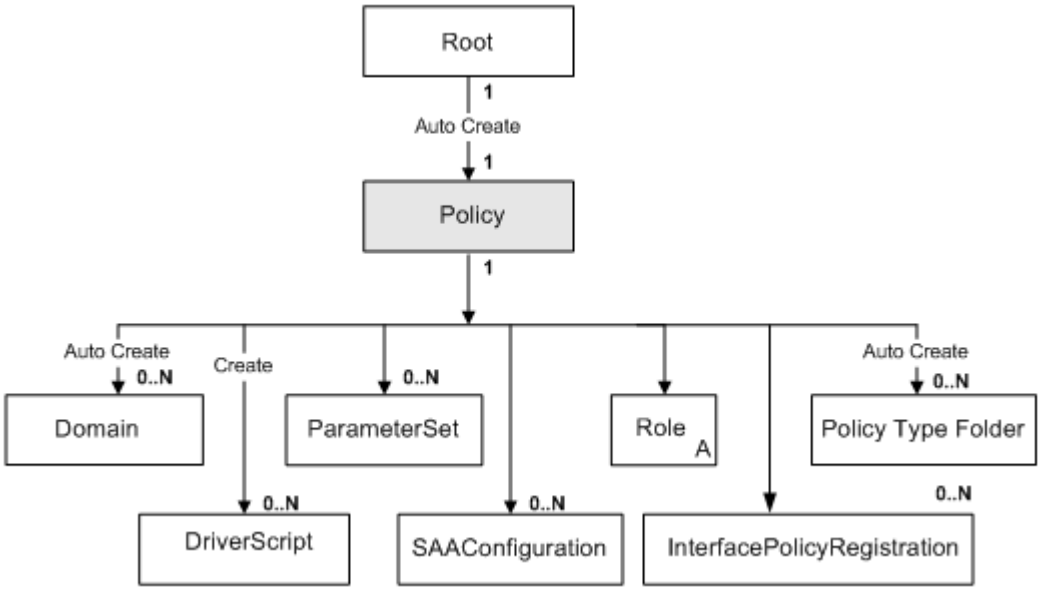


Table 4–8 describes the attributes for the Policy object.

Table 4–8 Policy Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Always “Policy”.

ParameterSet Object

ParameterSet objects represent a set of parameters that can be given different values when applied to different objects. ParameterSet objects define the names and data types of the parameters, but not the values.

ParameterSet objects have the following object inheritance:

```
ParameterSet.Object
```

You cannot create or modify ParameterSet objects through OIM.

Two ParameterSet objects are created automatically as children of the Policy object. These are called CollectorParameterSet and MeasurementParameterSet.

An example showing how parameter sets are applied to configuration targets is shown on "Applying Parameter Sets".

Figure 4–3 shows the ParameterSet object diagram.

Figure 4–3 The ParameterSet Object Diagram

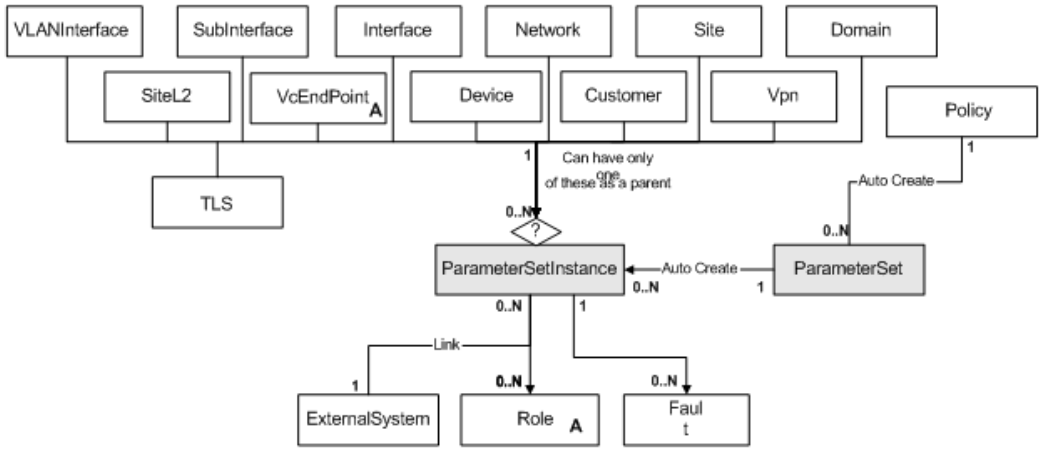


Table 4–9 describes the attributes for the ParameterSet object.

Table 4–9 ParameterSet Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of the ParameterSet.
Description	String	[no default]	RO	Description of the ParameterSet.

Table 4–9 (Cont.) ParameterSet Object Attributes

Attribute Name	Type	Default	Access	Explanation
Levels	U32	30	RO	The level at which the parameters apply. Bitwise value, 1 bit per target: Bit 0 = not used Bit 1 = Device Bit 2 = Interface Bit 3 = Subinterface Bit 4 = VCendpoint
Parameters	String	""	RO	The parameters that are defined by this ParameterSet. They will all have a value of "" as their values are only assigned when the ParameterSet is applied somewhere.
ExportedName	String	[no default]	RO	Name of parameter set as sent to device driver.

MeasurementParameterSet

The measurement parameters defined by this ParameterSet are described in Table 4–10.

Table 4–10 Measurement Parameters

Attribute Name	Type	Default	Access	Explanation
OCH_MeasureCBQoS	String	False	RW	Specifies whether to measure class based QoS.
OCH_MeasureCarQoSMIB	String	False	RW	Specifies whether to measure CAR QoS MIB.
OCH_MeasureJuniperCoSMIB	String	False	RW	Monitor Juniper CoS MIBs
OCH_MeasureMIB2Stats	String	False	RW	Specifies whether to measure SNMP MIB2.
OCH_NetflowEnabled	String	False	RW	Specifies whether to enable Netflow on the given object.
OCH_NetflowVersion	String	1	RW	Specifies which version of Netflow to use. 1=version 1, 2 =version 2, 3=Ag Only
OCH_NetflowAggregation	String	1	RW	Specifies which aggregation scheme to use for Netflow. 1=As, 2=Destination-prefix, 3=Prefix, 4=Protocol-port, 5=Source-prefix.
OCH_NetflowCacheSize	String	0	RW	Specifies the maximum number of entries in the cache. Range: 1024–524 288 0 = use default

Table 4–10 (Cont.) Measurement Parameters

Attribute Name	Type	Default	Access	Explanation
OCH_NetflowTimeoutActive	String	0	RW	Specifies the number of seconds before an inactive flow times out. Range 1–60. 0 = use default
OCH_NetflowTimeout Inactive	String	0	RW	Number of seconds flow is kept in cache. Range 10–600. 0 = use default

CollectorParameterSet

The collector parameters defined by this ParameterSet are described in Table 4–11.

Table 4–11 Collector Parameters

Attribute Name	Type	Default	Access	Explanation
OCH_CollectorName	String	[no default]	RO	Name of collector.
OCH_CollectorType	String	[no default]	RO	Type of collector system.
OCH_CollectorIpAddr1	String	[no default]	RO	Primary IP address of collector system.
OCH_CollectorIpAddr2	String	[no default]	RO	Secondary IP address of collector system.
OCH_CollectorPort1	String	[no default]	RO	Primary port number for collector system.
OCH_CollectorPort2	String	[no default]	RO	Secondary port number for collector system.
OCH_CollectorURL	String	[no default]	RO	URL for locating collector system.

ParameterSetInstance

A ParameterSetInstance object represents an instance of a ParameterSet that has been applied. It is a child of the relevant ParameterSet and the object to which the parameters are applied using the **use** command (see "The Use Command").

ParameterSetInstance objects have the following object inheritance:

```
ParameterSet.Object
```

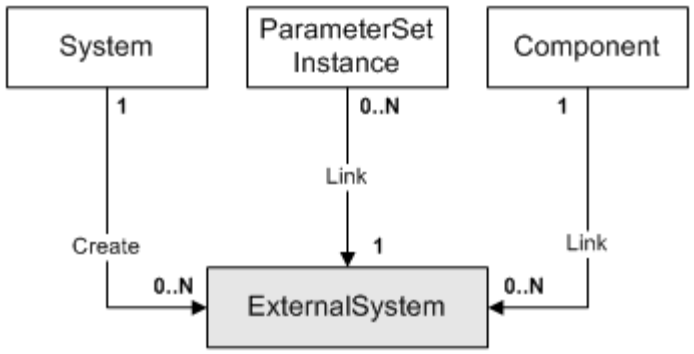
The ParameterSetInstance has the attributes defined by the ParameterSet object, but the parameters have actual values associated with them. Their types are correctly set, rather than just defined as strings, so for example, OCH_MeasureCBQoS is a Boolean value.

If the ParameterSetInstance is an instance of the MeasurementParameterSet, then the attributes of the ParameterSetInstance are modifiable by OIM.

If the ParameterSetInstance is an instance of the CollectorParameterSet, then the attributes of the ParameterSetInstance cannot be modified via OIM, but instead come directly from the ExternalSystem object that is a child of the ParameterSetInstance. The attributes of the ExternalSystem are modifiable, and modifying them also changes the parameter values of the ParameterSetInstance. For further information about ExternalSystem object, see "ExternalSystem Object".

Figure 4–4 shows the ParameterSetInstance object diagram.

Figure 4–4 ParameterSetInstance Object Diagram



An example showing how the parameter sets are applied to configuration targets is shown on "Applying Parameter Sets".

Table 4–12 describes the attributes for the ParameterSetInstance object.

Table 4–12 ParameterSetInstance Attributes

Attribute Name	Type	Default	Access	Explanation
Levels	U32	0	RW	Bitwise attribute. None = 0, VPN = 1, Device = 2, Interface = 4, SubInterface = 8, PVC = 16
Name	String	[no default]	RO	Name of the ParameterSetInstance.
Order	U32	0	RO	The order of the ParameterSetInstance.
Parameters	Varies	Varies	RO/RW	The parameters from the ParameterSet, with values which apply to the configuration target.

Domain Object

A policy domain is the logical organization for which policies and services can be defined, that is, all or part of a customer's network.

The Domain object has the following object inheritance:

```
Domain.Object
```

Figure 4–5 shows the Domain object diagram.

Figure 4–5 Domain Object Diagram

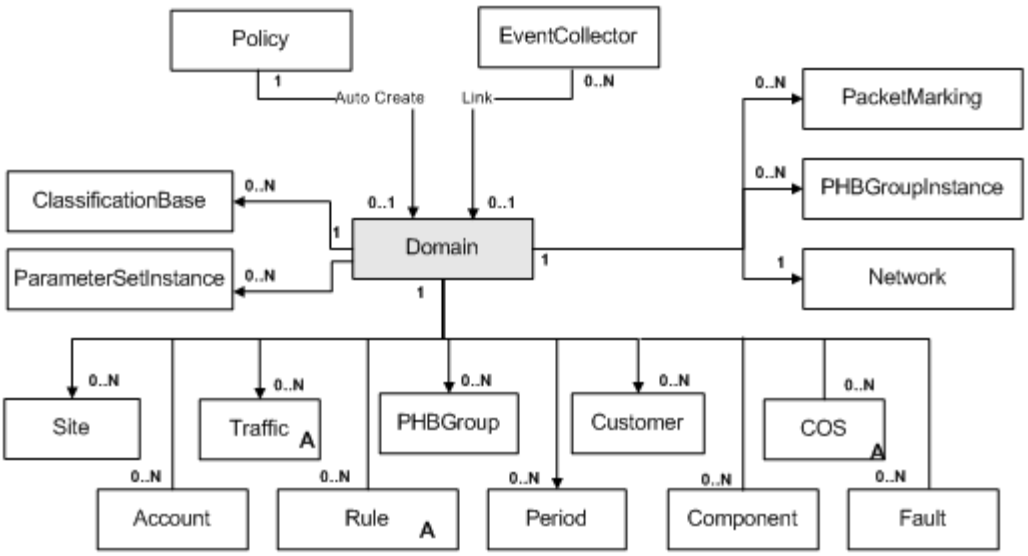


Figure 4–13 describes the attributes for the Domain object.

Table 4–13 Domain Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of the domain.
Remarks	String	[no default]	RW	Optional additional comments. Limit of 255 characters.
AsOverride	Boolean	True	RW	True = Set AS override for EBGP neighbors. False = Don't.
DomainAsOverride	Boolean	True	RW	True = Use the AsOverride value on all sites. False = Set AsOverride on each site.
AllowAsIn	U32	0	RW	Number of times the same AS can appear in the AS path list (0-10).
VrfTableLimit	U32	0	RW	Maximum number of routes allowed in a VRF (0=No limit).
VrfTableLimit Warning	U32	75	RW	Threshold at which warning is issued. Range: 1–101, where 1–100 = percentage warning 101 = warning only
BgpMd5Key	Encrypted string	[no default]	RW	BGP MD5 authentication key
PePeSendStandard Community	Boolean	True	RW	True = Use the PE-PE peering Standard send community tag. False = Don't.
PeCeSendStandard Community	Boolean	False	RW	True = Use the PE-CE peering Standard send community tag. False = Don't.

Table 4–13 (Cont.) Domain Object Attributes

Attribute Name	Type	Default	Access	Explanation
PeCeSendExtended Community	Boolean	False	RW	True = Use the PE-CE peering Extended send community tag. False = Don't.
LoadPolicy	String	unset	RW	Loads a policy file into the object_model.
UseLoopback	String	0	RW	The Loopback ID value is used to create a loopback interface name by appending it to the name 'loopback'. For example, if the Loopback ID is 0, the loopback interface name created is 'loopback0'. When a device in this domain is discovered, a check is made to see if a loopback interface matching this text string exists. If it does, the IP address of the loopback interface is stored with the device information.

InterfacePolicyRegistration

The interface policy registration informs the IP Service Activator core about a generic policy and its abilities, and is restricted to the interface management scope. Within this scope, the information about a policy is characterized by:

- Operation: What is the name of the policy? What does this policy do - Create or Decorate? How should this policy be represented in the GUI menus?
- Context: What kind of device does this policy apply to? At what level can this policy be made available - device, controller, interface or subinterface? If a controller, interface or subinterface is the level chosen, what kind of controller, interface or subinterface should it be?
- Creation Template: If the policy type is Create, what kind of interface or subinterface does it create? Is there any pattern that its name should follow? What kind of capabilities should that object have upon creation?

Creation capabilities apply only when the object being created is an interface. In that case, all the interface level capabilities have to be specified (linked) - interface caps, subinterface caps and VC caps. Note that you must not modify a caps object linked to a registration. It might be in use by other interfaces - once a caps object is created under the Topology, it is immediately shareable. Instead, create a new caps object, unlink the old one and link in the new one. Or just repeat the interface linkage with another interface that has the desired caps.

A given policy may be registered many times depending upon how flexible it is. For instance, a policy may be able to create a subinterface or decorate an existing one. In such a case, you would register this policy twice - once each for Create and Decorate. Once an interface management generic policy has been registered appropriately, it is ready for use.

Once a given registration is in use, the management operations that can be performed on it are limited. This is to ensure we do not orphan the current users or create a mismatch between the current users and post-modification future users. In-use registrations cannot be deleted. They can be modified, but only certain attributes can change:

- They can be disabled. A disabled registration no longer appears on any GUI menus and cannot be used anew - the existing usages remain unimpacted.
- Their name can change.

- Their menu text can change.

In order to create an interface (subinterface), the user must set their path to the Device (Interface). They have to then create a stub object: "create Interface:SomeName", or "create SubInterface:SomeName". Then they have to link the desired registration (child) to the stub object (parent). It is recommended they commit at this point so their EOM is updated with the new object. They have to then link the desired role (child) to the stub object. Under the stub object, they will find the generic policy instance. They have to modify its payload attributes with the appropriate XML string. Now, a commit will complete the creation operation.

To configure an object, you must simply link the desired registration (child) to the desired object (parent). It is recommended they commit at this point so their EOM is updated with the new objects. The desired object must be linked to an appropriate role. Under that desired object, they will find the generic policy instance. They have to modify its payload attributes with the appropriate XML string. Now, a commit will complete the configure operation.

To delete a created interface or subinterface, the user must simply delete the object.

To remove the configuration of a previously configured object, the user must simply unlink the applied registration from the object.

Figure 4–6 shows the InterfacePolicyRegistration object diagram.

Figure 4–6 InterfacePolicyRegistration Object Diagram

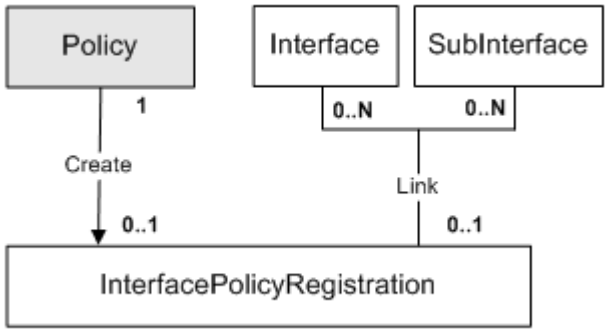


Table 4–14 describes the attributes for the InterfacePolicyRegistration object.

Table 4–14 InterfacePolicyRegistrationAttributes

Attribute Name	Type	Default	Access	Explanation
Context	Enum	Device	RW	The kind of object you will apply this registration on - either a device, controller, interface or sub-interface. 0 = "Device" 1 = "Interface" 2 = "SubInterface" 3 = "Controller"
ContextDefault	Boolean	False	RW	Context attribute is used only when ContextDefault is set to false.

Table 4–14 (Cont.) InterfacePolicyRegistrationAttributes

Attribute Name	Type	Default	Access	Explanation
ContextPattern	String		RW	Used only if the Context is Controller, Interface or SubInterface. The context matching is extended to additionally require that the context name begins with this pattern.
ContextSnmpIfType	U32	0	RW	The interface type. Valid for all Applied Contexts except for device. For example, if you are creating a sub-interface on an interface that has an ifType of "32", then the SNMP ifType is "32".
ContextSnmpIfTypeList	String		RW	Comma separated list of SNMP ifType of the interfaces or subinterfaces where this policy can be chosen. It is used only if Context is Interface or SubInterface and if ContextDefault is not set.
CreationPrefix	String		RW	This value pre-populates the interface properties page when you use this registered policy and allows IP Service Activator to validate the supplied interface or sub-interface name during creation.
CreationSnmpIfType	U32	6	RW	This value pre-populates the interface properties page with SNMP ifType.
CreationUseParentPrefix	Boolean	False	RW	Valid for sub-interface creation only. This value indicates whether the registered policy will get the prefix from the parent interface.
Enabled	Boolean	True	RW	Whether the registration is operational.
GenericPolicyTypeName	String		RW	Generic policy type
MenuText	String		RW	GUI: The text that will display in the right-click drop-down menu at the point where you wish to apply interface management policies.
Name	String		RO	Name for the interface policy registration
PolicyType	Enum	Create	RW	Identifies the interface policy type being registered.
SnmpIfSpeed	U32	0	RW	Default bandwidth for created interface or sub-interface in kbps. This value is only for validation purposes. The actual speed depends on the given configuration policy.

Table 4–14 (Cont.) InterfacePolicyRegistrationAttributes

Attribute Name	Type	Default	Access	Explanation
VendorEnterpriseId	U32	0	RW	A vendor-specific and series-specific ID. Refers to the kind of device you are applying the registration to. For example, the SNMP Enterprise ID for Cisco is "9" and for Juniper it is "2636".

Customer Object

A Customer object represents the concept of a customer, to which VPNs and sites are linked.

Customer objects have the following object inheritance:

```
Customer.Object
```

Figure 4–7 shows the Customer object diagram.

Figure 4–7 Customer Object Diagram

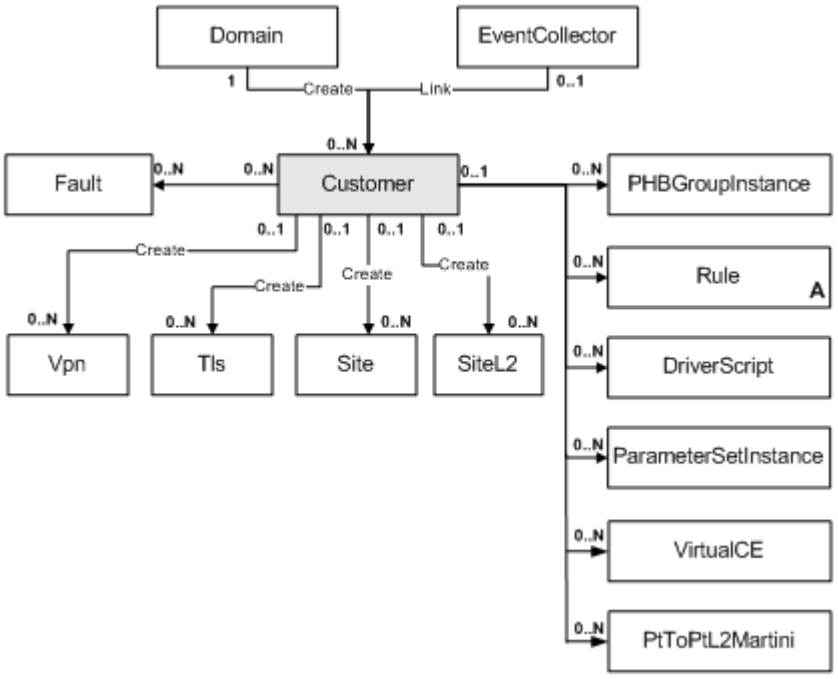


Table 4–15 describes the attributes for the Customer object.

Table 4–15 Customer Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the customer.
Remarks	String	[no default]	RW	Optional additional comments. Limit of 255 characters.

Table 4–15 (Cont.) Customer Object Attributes

Attribute Name	Type	Default	Access	Explanation
Reference	String	[no default]	RW	Customer reference. No format is imposed.

SiteFolder

The SiteFolder object defines a site folder used to contain site objects and site subfolders. A SiteFolder can be a child of Customer or a child of another SiteFolder object.

Figure 4–8 shows the SiteFolder object diagram.

Figure 4–8 SiteFolder Object Diagram

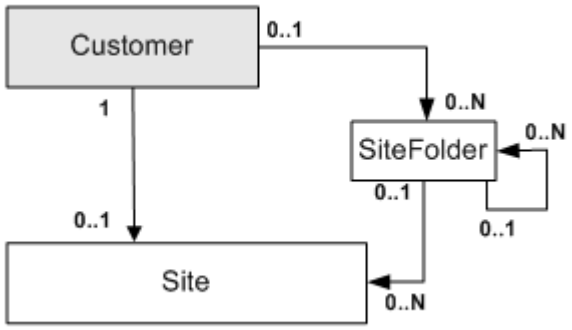


Table 4–16 describe the attributes for the SiteFolder object.

Table 4–16 SiteFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
AccountRef	String		RW	Text field to store customer reference information. Note: No inheritance is implied between sites in the site folder(s) and the Account Ref field value in the site folder.
Remarks	String		RW	Optional additional information about the site. This is a free-format text string. It is for information only and is not used by IP Service Activator. Limit of 255 characters.
Name	String		RW	

CustomerFolder Object

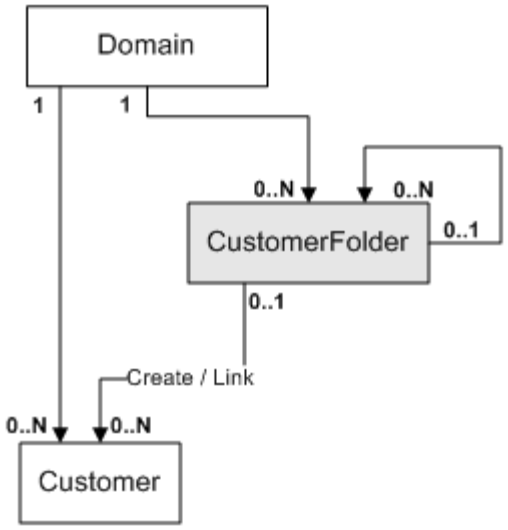
A CustomerFolder object represents a folder which contains customer objects, or other CustomerFolder objects, for purposes of organization within the GUI.

CustomerFolder objects have the following object inheritance:

```
CustomerFolder.Object
```

Figure 4–9 shows the CustomerFolder object diagram.

Figure 4–9 CustomerFolder Object Diagram



A Customer always has either 1 or 2 parents and is always linked to its parent Domain. It may be linked to zero or one parent CustomerFolders. In the IP Service Activator GUI, if a Customer has 2 parents, it will always be displayed under the CustomerFolder, not under the Customers folder. If a customer is created under a folder, it is automatically linked to its parent Domain.

A Customer folder is a child of either another Customer folder or the Domain, but not both.

Table 4–17 describes the attributes for the CustomerFolder object.

Table 4–17 CustomerFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the customer folder.
Remarks	String	[no default]	RW	Optional additional comments. Limit of 255 characters.

Account Object

An account is a source or destination point in the network to which policy rules can be applied. An account in this context can be one of the following:

- A user, identified by a name and an IP address
- A host computer, identified by an IP address
- A subnet

Account objects have the following object inheritance:

```
Account.Object
```

Accounts can be organized into a hierarchical structure of groups for organizational purposes or to represent the structure of a company.

- Accounts must be set up via the IP Service Activator user interface, but existing accounts can be browsed in the EOM.
- In the EOM, Account objects are not strictly required for the correct operation of policy objects; use of Accounts is therefore limited.
- Existing accounts may be referred to by name when manipulating Rule objects.
- Rules do not require host/subnet accounts to be entered; IP addresses may be input directly.

Figure 4–10 shows the Account object diagram.

Figure 4–10 Account Object Diagram

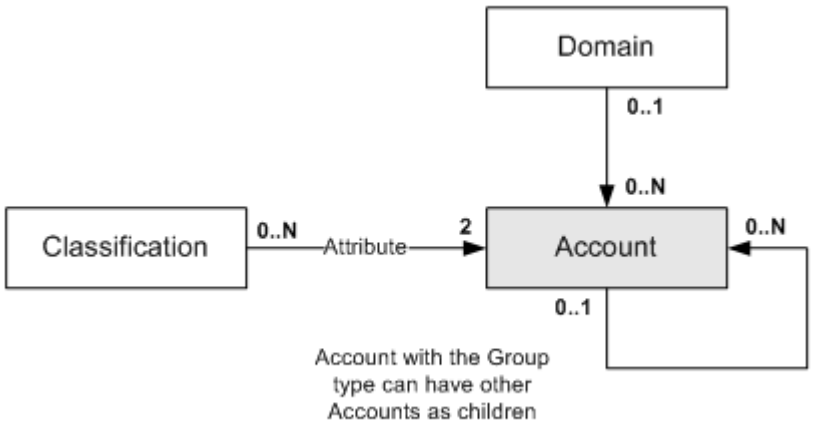


Table 4–18 describes the attributes for the Account object.

Table 4–18 Account Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of the account.
Remarks	String	[no default]	RO	Optional additional comments. Limit of 255 characters.
Type	String	"User"	RO	Type of account; one of: ■ Group: an administrative grouping ■ User: A named user and IP address ■ Host: A host by IP address ■ Subnet: A subnet

VPN Object

A virtual private network, defined by a set of sites (PE interfaces or CE routers) at the edge of the core network cloud. A VPN is a general concept and is independent of the various technologies that may be used to provide the privacy and/or routing independence.

The VPN object has the following object inheritance:

```
Vpn.Object
```

Figure 4–11 shows the VPN object diagram.

Figure 4–11 VPN Object Diagram

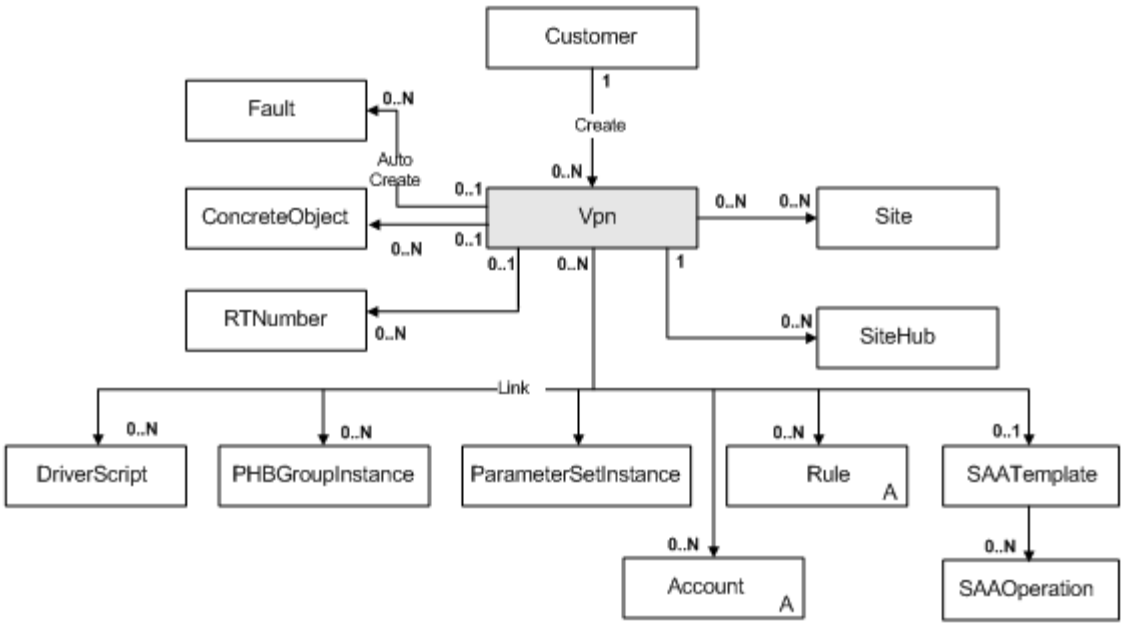


Table 4–19 describes the attributes for the VPN object.

Table 4–19 VPN Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the VPN.
Description	String	[no default]	RW	Free-format comments about the VPN. Limit of 255 characters.
Type	Enum	0		Type of VPN: 0 = None 1 = MplsVpn
Level	U32	4	RW	Priority level of the VPN, in the range 0–7. This value is only used if a site is included in more than one VPN, which may result in conflict. The VPN with the lowest level number will take precedence.
MplsVpnType	Enum	0	RW	Topology of MPLS VPN: 0 = FullMesh 1 = HubAndSpoke 2 = Management
VpnId	U32	0	RO	Unique number of VPN relative to its customer.

Table 4–19 (Cont.) VPN Object Attributes

Attribute Name	Type	Default	Access	Explanation
Context	String	[no default]	RW	Local context for driver scripts applied at network level (max 512 bytes).
UseOneRdPer Vpn	Boolean	False	RW	True = Apply the same VRF table name and RD number to all sites that participate in this VPN False = Generate a site-specific VRF table name and RD number for each site that participates in the VPN
InstallDhcp	Boolean	False	RW	True = Install DHCP support on the VRFs. False = Do not install DHCP support on VRFs.
PrimaryDhcpIpAddr	IPAddress	0.0.0.0	RW	Primary DHCP Server
SecondaryDhcpIpAddr	IPAddress	0.0.0.0	RW	Secondary DHCP Server
EbgpMd5Key	Encrypted string	[no default]	RW	BGP MD5 authentication key [no default]=not used.
OspfMd5Key	Encrypted string	[no default]	RW	OSPF MD5 authentication key [no default]=not used.
Multipath	Boolean	False	RCW	Applies protocol-independent load balancing to the default routing table for that routing instance (applicable for routers running JUNOS)
VpnUnequalCost	Boolean	False	RCW	Speficies what routes to consider (applicable for routers running JUNOS)
EqualExternalInternal	Boolean	False	RCW	Specifies that both external and internal BGP paths can be selected for multipath. (applicable for routers running JUNOS)

The attributes described in Table 4–20 only apply if UseOneRdPerVpn is set to True, i.e. VRF tables and RD numbers are defined once per VPN, rather than separately for each site (see also "Site objects").

Table 4–20 VPN Attributes if UseOneRdPerVpn is Set to True

Attribute Name	Type	Default	Access	Explanation
ForceVrfInstall	Boolean	True	RW	True = VRF tables on corresponding interfaces must be installed and cannot be merged into other tables. False= VRF tables can be merged into other tables.

Table 4–20 (Cont.) VPN Attributes if UseOneRdPerVpn is Set to True

Attribute Name	Type	Default	Access	Explanation
ShareableVrf	Boolean	False	RW	True = Other tables can be merged into this VRF table. False = Other tables cannot be merged into this VRF table.
OverrideVrf Table Limit	Boolean	False	RW	True = Use site-specific settings for VRF table limits False = Use domain defaults for VRF table limits
VrfTableLimit	U32	0	RW	Maximum number of routes allowed in a VRF (0=No limit).
VrfTableLimit Warning	U32	0	RW	Percentage at which to warn of VRF table limits being exceeded. Range: 1–101 1–100 = percentage of VrfTableLimit reached warning. 101 = warning when VrfTableLimit reached.
VrfTableName	String	[no default]	RW	The name of the VRF routing table.
RDHighOrder	U32	0	RW	The top 32 bits of the Route Descriptor value.
RDLowOrder	U32	0	RW	The bottom 32 bits of the Route Descriptor value.
VrfImport	U32	1	RW	The number of device redundant path configurations. Range is 2^{16} plus the default.
UseVrfImport	Boolean	True	RW	True = Use VRF import False = Do not use VRF import
EBgpMaxPaths	U32	1	RW	Allows the specification of the maximum number of parallel EBGP routes that can be installed on the device. This corresponds to the Cisco maximum-paths command. Range: 1–16.
IBgpMaxPaths	U32	4	RW	Allows the specification of the maximum number of parallel EBGP routes that can be installed on the device. This corresponds to the Cisco maximum-paths command. Range: 1–16.

Table 4–20 (Cont.) VPN Attributes if UseOneRdPerVpn is Set to True

Attribute Name	Type	Default	Access	Explanation
EIBgpMaxPaths	U32	1	RW	Allows the specification of the maximum number of parallel EBGP and IBGP routes that can be installed on the device. This corresponds to the Cisco maximum-paths eibgp command. Range: 1–16.
EigrpMaxPaths	U32	1	RW	Allows the specification of the maximum number of parallel EIGRP routes that can be installed on the device. This corresponds to the Cisco maximum-paths eigrp command. Range: 1–16 for IOS 12.3(2)T and later 12.3(T), and from 1–6 in earlier versions. The default value is 4.
EigrpMd5KeyChainRef	String	--	RW	Specify the key chain name to use with MD5 Authentication for EIGRP.
IBgpUnequalCost	Boolean	False	RW	Allows unequal cost load balancing by selecting iBGP paths that do not have an equal cost.
OspfMd5AreaLevelAuth	Boolean	False	RW	Enables MD5 key authentication for OSPF for the VPN.
UseVrfLabel	Boolean	False	RW	Enables Juniper vrf-table-label support. When this field is set to true, the inner (VPN) label of a packet is removed as it arrives at a VRF so that it can be processed based on the contents of its IP header. When this field is set to false, incoming packets are mapped directly onto an outgoing (CE-facing) interface based on the inner VPN label.
VrfDesc	String	--	RW	Allows a VRF table route to be advertised to other PE routers only if its prefix matches one of those specified in the export map. The export map tags exported routes with the RT number of each site that needs to receive those routes. Export maps must be manually pre-configured on the PE router.

Table 4–20 (Cont.) VPN Attributes if UseOneRdPerVpn is Set to True

Attribute Name	Type	Default	Access	Explanation
VrfExportFilter	String	--	RW	Allows a VRF table route to be advertised to other PE routers only if its prefix matches one of those specified in the export map. The export map tags exported routes with the RT number of each site that needs to receive those routes. Export maps must be manually pre-configured on the PE router.
VrfImportFilter	String	--	RW	A VRF Import Map allows the site to selectively import routes learned elsewhere.
CreateDefaultRTNumbers	Boolean	True	RC	Applies only when the VPN is created. If set to True, IP Service Activator automatically creates two route targets (RTNumber objects) for the newly created VPN. If set to False, you must create one or more route targets (RTNumber objects) for the newly created VPN.

PtToPtL2Martini Object

A Layer 2 Martini virtual private network, defined by endpoints (PE interfaces or CE routers) at the edge of the core network cloud and encapsulating various types of data across the Martini VPN tunnel.

The PtToPtL2Martini object has the following object inheritance:

```
L2PtToPtMartini.Object
```

Figure 4–12 shows the PtToPtL2Martini object diagram.

Figure 4–12 PtToPtL2Martini Object Diagram

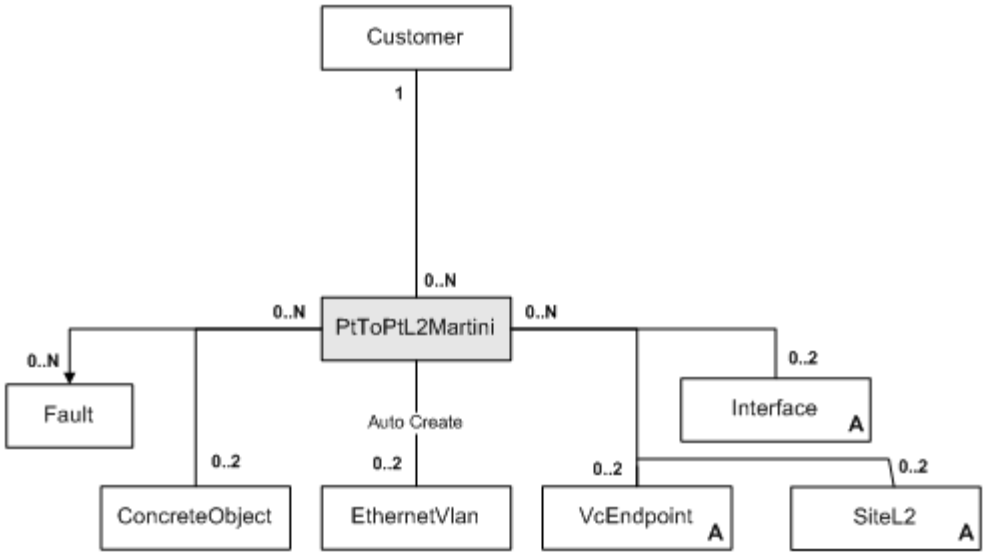


Table 4–21 describes the attributes for the PtToPtL2Martini object.

Table 4–21 PtToPtL2Martini Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Simple accessory to this object.
Remarks	String		RW	General comments. Limit of 255 characters.
ConnectionType	Enum	0	RW	The type of Martini connection to be made: ATM_AAL5 = 0 ATM_Cell = 1 Frame = 2 Ethernet = 3 Ethernet_VLAN = 4
MartiniVcId	U32	1	RW	Range: 1–0xFFFFFFFF
ActualMartiniVcId	U32	0	RO	Value actually in use, user-defined or generated
GenerateIdentifier	Boolean	False	RW	Allows the Martini site to be created if set to True. Note: When modifying bGenerateIdentifier to False you should also modify the MartiniVcId in the same transaction, as the default is '1'.

TLS Object

A Transparent LAN Service is a service used to connect together separate LAN segments via an MPLS cloud and make them appear as if they were forming a unique

VLAN. TLS objects can be used to connect distinct physical ports or specific VLANs associated with physical ports. In the first case, the ports are known as “host ports”, whereas in the second case they are called “trunk ports” (in both case, ports are a synonym for the Interface objects).

The TLS object has the following object inheritance:

```
Tls.Object
```

Note that policy (rules, PHB groups and driver groups) applied on the TLS or higher up in the inheritance hierarchy (for example on the customer) is not inherited through TLS objects down to the specific ports.

If a TLS has a ServiceType attribute of PortAndVlan, the range of VLAN IDs specified on the TLS will be used to set up the TLS customer profile on the PE port. The actual VLANs created on the PE port and on the MTU ports can be specified on the site itself (but must be a subset of the VLAN IDs specified on the TLS). If no VLAN IDs and no encapsulation are specified on the L2Site object, all the VLANs that are part of the TLS range will be created on the PE and MTU in the site.

The system validates that on a CE/MTU the same VLAN is not used in the context of different TLS services, to avoid cross communications between the TLSs. It is assumed that on the PE, VLANs created in the context of different PortAndVlan based TLSs will be prevented from communicating by the customer profile.

Figure 4–13 shows the TLS object diagram.

Figure 4–13 TLS Object Diagram

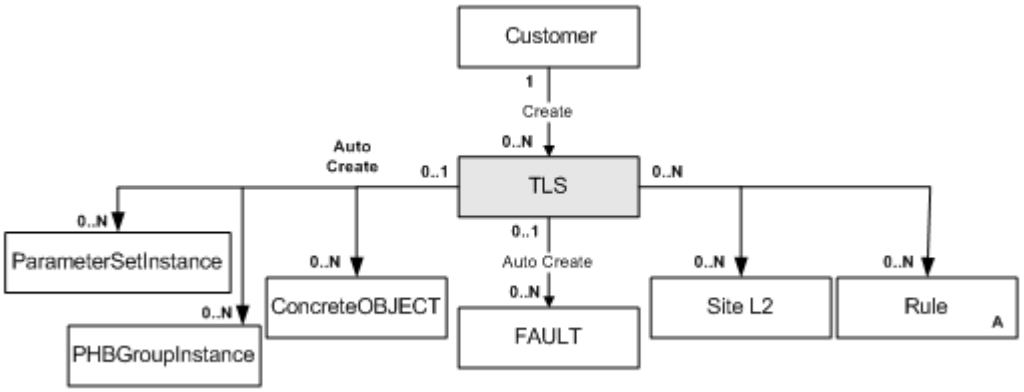


Table 4–22 describes the attributes for the TLS object.

Table 4–22 TLS Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of Layer 2 VPN.
Description	String	[no default]	RW	Free-format description of the Layer 2 VPN.

Table 4–22 (Cont.) TLS Object Attributes

Attribute Name	Type	Default	Access	Explanation
Topology	Enum	0	RW	VPN topology. At present, only Full Mesh is supported. 0 = FullMesh 1 = HubAndSpoke 2 = Management
ServiceType	Enum	0	RW	Defines how access to the TLS is managed 0 = Port Based 1 = PortAndVlanBased
ProfileVlanIds	String	[no default]	RW	If ServiceType = 1, specifies which VLANs should be made part of the customer profile. Can be a single VLAN ID, or several comma- separated IDs, and can include ranges (e.g. 4, 10-15)
PortsConfiguration	Enum	0	RW	If ServiceType = 0, all PE ports in all sites share the same configuration: 0 = OnlyUntaggedFrames Accept only untagged frames (access port) 1 = OnlyTaggedFrames Accept only pre-tagged frames (trunk port)
Conflict	Boolean	False	RO	True = TLS is failing. False = TLS is not in error.
GenerateVCId	Boolean	True	RW	autogeneration of the VC ID.
Stacked	Enum	Unstacked	RW	To select a Stacked VLAN or Unstacked. Use Stacked when the packets coming into the site from the CE are already tagged with a customer-specific VLAN ID. An additional tag is added as the packets move out of the site to the PE. Valid entries are: Stacked, Unstacked.
StackedVlanTag	U32	0	RW	Provide the VLAN ID(s) that packets are to be tagged with.

SAATemplate Object

An SAATemplate object is a parent object which groups a number of SAAOperation objects.

The SAATemplate object has the following object inheritance:

```
SAATemplate.Object
```

Figure 4–14 shows the SAATemplate object diagram.

Figure 4–14 SAATemplate Object Diagram

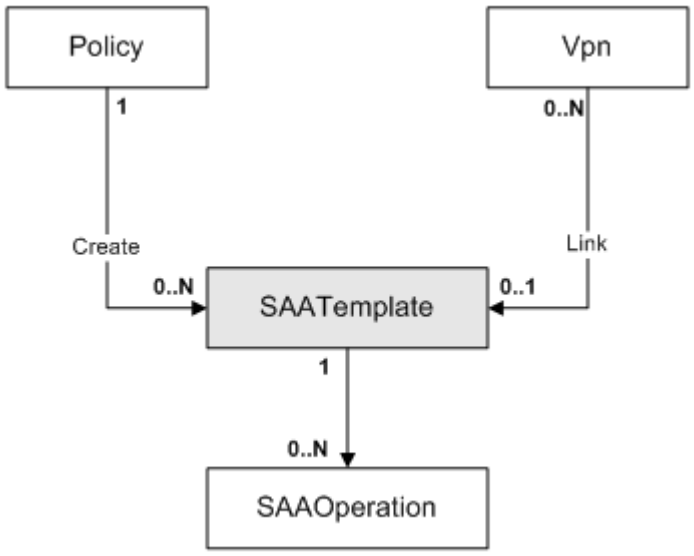


Table 4–23 describes the attributes for the SAATemplate object.

Table 4–23 SAATemplate Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the SAA template.
Remarks	String	[no default]	RW	Optional additional comments. Limit of 255 characters.
DeviceBits	U32 [6-11]	10	RW	Defines how many devices can be configured per VPN ID (allowing up to 1024 devices per SAA VPN).
TosBits	U32 [1-3]	2	RW	Defines how many probes (SAA operations) can be configured (up to 64).
TypeBits	U32 [1-3]	2	RW	Defines how many measurement types can be configured per VPN (up to 8).

Note: The combined value of the DeviceBits, TosBits and TypeBits attributes must be exactly 14.

SAAOperation Object

An object that represents the parameters that will be used to configure an SAA operation and where the operation will be configured; that is, on one (half duplex) or both (full duplex) devices for each tested connection in a VPN.

The SAAOperation object has the following object inheritance:

```
SAAOperation.SAATemplate.Object
```

See Figure 4–14 for the SAAOperation object diagram.

Table 4–24 describes the attributes for the SAAOperation object.

Table 4–24 SAAOperation Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of the SAA operation object.
Type	Enum	0	RW	Type of operation: 0 = IcmpEcho 1 = TcpConnect 2 = UdpEcho 3 = Jitter
Duplex	Enum	0	RW	The test performed on a connection: 0 = HalfDuplex 1 = FullDuplex
Period	U32	0xFFFFFFFF	RW	Frequency with which a measurement is performed (seconds). Range: 0–604800 Default value indicates no specific period will be used.
Timeout	U32	0xFFFFFFFF	RW	Timeout in seconds. Must not be greater than frequency. Range 0–604 800 000 Default value indicates no specific timeout will be used.
Lifetime	U32	0xFFFFFFFF	RW	Lifetime of the operation defined in seconds. A value of –1 indicates the operation is active forever. Range: 0–2 147 483 647
RisingThreshold	U32	0xFFFFFFFF	RW	Sets the rising threshold that generates a reaction event and stores history information for the operation. Defined in milliseconds. Frequency with which a measurement is performed (seconds). Default value indicates no specific period will be used. Range: 0–2 147 483 647
FallingThreshold	U32	0xFFFFFFFF	RW	Falling threshold in milliseconds. Default value indicates no specific period will be used. Range: 0–2 147 483 647
ThresholdType	Enum	1	RW	Algorithm used to calculate the passing of thresholds. 0 = Never 1 = Immediate 2 = Consecutive 3 = XofY 4 = Average

Table 4–24 (Cont.) SAAOperation Attributes

Attribute Name	Type	Default	Access	Explanation
Consecutive Occurrences	U32	0	RW	The number of consecutive threshold violations that trigger the action defined by the ActionType attribute. 0 = Use default. Range: 0–16
Xofy_x	U32	0	RW	Value of X when ThresholdType is XofY. 0 = Use default. Range: 0–16
Xofy_y	U32	0	RW	Value of Y when ThresholdType is XofY. 0 = Use default. Range: 0–16
Average	U32	0	RW	Parameter for 'Average' ThresholdType. Range: 0–16
ActionType	U32	0	RW	Bitwise value, indicating actions that occur when the threshold is passed. 0 = None 1 = Trap 2 = Nmvt 4 = Trigger
DsCodepoint	U32	0	RO	The IP ToS byte for request packets. Applies to ICMP Echo, UDP Echo and Jitter operations.
DestPort	U32	0	RW	Destination port number. 0 = use default. The following defaults apply: TcpConnect: 23 UdpEcho: 7 Jitter: 8000
SourcePort	U32	0	RW	Source port number. 0 = use default. The following defaults apply: TcpConnect: 23 UdpEcho: 7 Jitter: 8000
RequestSize	U32	28-Icmp 1-Tcp 16-Udp 32-Jitter 28-Http 1-Dns	RW	Sets the protocol data size in the payload of the operation's request packet. 0xFFFFFFFF = Use default. Valid sizes are: IcmpEcho: >0–65535 TcpConnect: >=0–65535 UdpEcho: 4-8192 Jitter: 16-1500 Http: >=0 Dns: >=0

Table 4–24 (Cont.) SAAOperation Attributes

Attribute Name	Type	Default	Access	Explanation
EnableControl	Boolean	False	RW	Enable/disable control message sent to destination port. Applies to TcpConnect, UdpEcho and Jitter.
PacketsInSequence	U32	10	RW	Number of packets in sequence. Range: 1–60000
PacketInterval	U32	20	RW	Inter-packet interval. Range: 1–60000 ms
EnableErrorChecking	Boolean	False	RW	True=Enables error verification checking.
EnableConnect Checking	Boolean	False	RW	True=Enables checks for connection loss in connection-oriented protocols.
EnableTimeout Checking	Boolean	False	RW	True=Enables checks for RTR operation timeouts.
HistoryLives	U32	0	RW	Number of entries (lives) that are stored in the history table for a given operation. Range: 0–25
HistoryFilter	Enum	0 – None	RW	Defines the type of history information that will be collected in the history table: 0 = None 1 = All 2 = OverThreshold 3 = Failures
HistoryBuckets	U32	15	RW	Specifies how many data points from which to record data for a given operation. Range: 1–60
SourcePortSpecified	Boolean	False	RW	Enables or disables the use of a SourcePort attribute.
TagName	String	"<Default>"	RW	Tag for the SAA operation. Default value configures tag name as the customer name specified in UI.

RtNumber Object

Represents a Route Target. A user can create any number of RTs, and for each one select under what circumstances it is to be exported or imported.

The RtNumber object has the following object inheritance:

```
RtNumber.Object
```

Figure 4–15 shows the RtNumber object diagram.

Figure 4–15 RtNumber Object Diagram

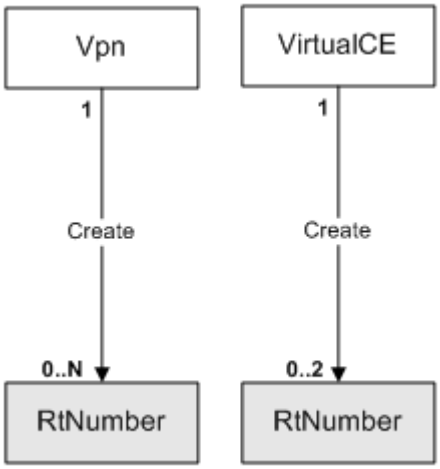


Table 4–25 describes the attributes for the RtNumber object.

Table 4–25 RtNumber Object Attributes

Attribute Name	Type	Default	Access	Explanation
RtHighOrder	U32	0	RW	Route Target number – high order value. If both RtHighOrder and RtLowOrder are set to 0, then the system default is used (the Domain ASN).
RtLowOrder	U32	0	RW	Route Target number – low order value. If both RtHighOrder and RtLowOrder are set to 0, then the system default is used (the ObjectId * 2).
HubBehaviour	Enum	0	RW	0 = None 1 = Import 2 = Export 3 = ImportExport
SpokeBehaviour	Enum	0	RW	0 = None 1 = Import 2 = Export 3 = ImportExport
MeshBehaviour	Enum	0	RW	0 = None 1 = Import 2 = Export 3 = ImportExport **Note:** Values 0 (None) and 3 (ImportExport) are not supported for Virtual CEs. The complete set of route target restrictions for Virtual CEs follows: ▪ mesh ▪ zero or one import ▪ zero or one export

Table 4–25 (Cont.) RtNumber Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the RtNumber object.
VrfTarget	Boolean	False	RCW	True = The Juniper cartridge generates VPN configuration using the VRF Target format. One each of Import, Export, and ImportExport can be created on Juniper devices. False = flag is disabled (default). The Juniper cartridge generates policy-based VPN configuration.
AddressFamily	Enum	2	RW	Address family for route target. Default is ipv4_ipv6 0 = Ipv4 1 = Ipv6 2 = Ipv4_Ipv6

RtNumber for Sites and VPNs

Two RtNumber objects are automatically created whenever a VPN is created with the attribute CreateDefaultRTNumbers set to its default value of True. The attributes of the RT numbers are set according to whether the VPN's MplsVpnType attribute is HubAndSpoke, or FullMesh. The RT numbers represent the default VPN behavior, but may be altered by the user for different behavior. For example, in a hub and spoke VPN with several hubs, the hubs can be set to meshed or non-meshed by altering the RT number object's attributes.

Each RtNnumber object is given RtLowOrder and RtHighOrder values which are unique to the system. RtNumber objects also have three other attributes: HubBehaviour, SpokeBehaviour, and MeshBehaviour which can take one of four values: Import, Export, ImportExport, or None.

By default, one of the RtNumber objects has HubBehaviour set to ImportExport, SpokeBehaviour set to Import, and MeshBehaviour set to ImportExport. The other RtNumber has HubBehaviour set to Import, SpokeBehaviour set to Export, and MeshBehaviour set to None.

A site in a VPN can see another site if it imports an RtNumber that the other site exports.

If the VPN is fully meshed, then the MeshBehaviour attribute is used from the RtNumber objects. In the default case, the MeshBehaviour is to import and export the first RtNumber, so all the sites import and export the same RtNumber ensuring that all the sites are meshed.

If the VPN is hub and spoke, the hub sites use the HubBehaviour, and the Spoke sites use the SpokeBehaviour. In the default case, the hub sites import and export Rt1. The spoke sites import Rt1, so the spokes can see the sites. The hub sites also import Rt2, and the spoke sites export Rt2, so the hubs can see the sites. The spokes export Rt2, but do not import Rt2, so the spokes cannot see other spokes. The hubs export Rt1, and import Rt1 so the hubs can see other hubs.

One common customization is to make the hubs non-meshed so that hubs cannot see other hubs. To achieve that, change the hub behavior of Rt1 from ImportExport to Export. This means that the hubs do not import the same number that they export, so the hubs will not be able to see each other.

RtNumber for Virtual CEs

An RtNumber (RT) can be provisioned under a Virtual CE site. This functionality is enabled in the Virtual CE.

Site objects

A Site object has two types: VPN or Virtual CE.

The Site object has the following object inheritance:

```
Site.Object
```

A Site object of type VPN represents a physical Site defined by one or more devices and interfaces. PHB groups and rules can be defined at a site level to apply to devices and interfaces within the site or closely related (such as an interface providing the VPN connection). Rules applied at site level are lower priority than rules defined at the device level.

A Site object of type Virtual CE represents a physical Site defined by one or more interfaces from a single CE device. PHB groups and rules can be defined at a site level to apply to interfaces within the site or closely related (such as an interface providing the VPN connection). Rules applied at site level are lower priority than rules defined at the device level.

Figure 4–16 shows the Site object diagram.

Figure 4–16 Site Object Diagram

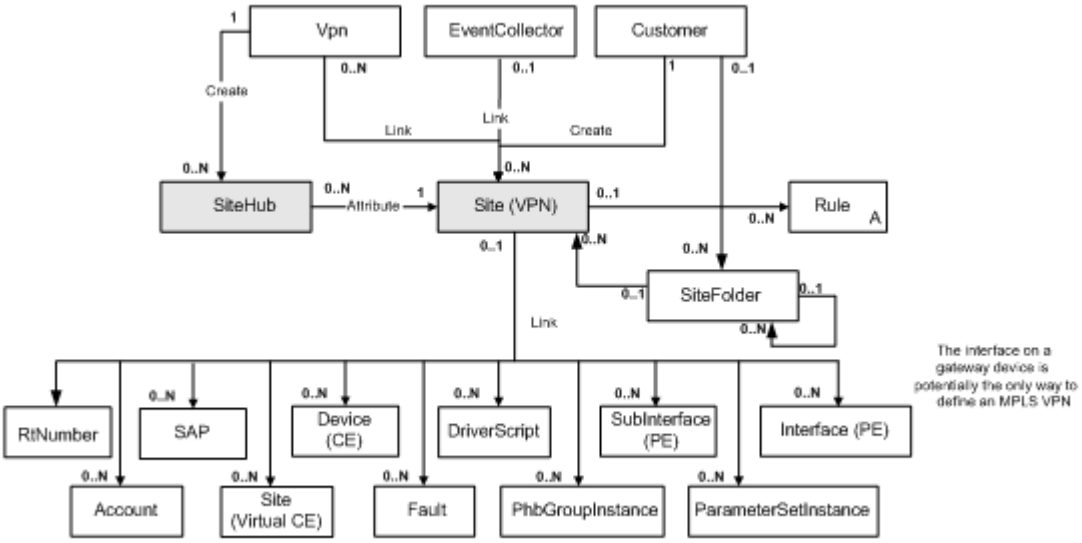


Table 4–26 describes attribute common to both VirtualCE and VPN Site objects.

Table 4–26 Attributes Common to VirtualCE and VPN Site Objects

Attribute Name	Type	Default	Access	Explanation
Type	Enum	VPN	RC	Type of site: 0 = VPN 1 = VirtualCE Type is defined when the object is created. You cannot convert one type to the other.
Name	String	[no default]	RW	Name of the site.
Remarks	String	[no default]	RW	Optional additional comments. Limit of 255 characters.
Contact	String	[no default]	RW	Name of primary contact at site.
Address	String	[no default]	RW	Postal address of site.
Telephone	String	[no default]	RW	Site phone number for contact.
Fax	String	[no default]	RW	Site fax number for contact.
Email	String	[no default]	RW	Site e-mail address for contact.
AccountRef	String	[no default]	RW	Unique account number.
SiteOfOrigin	U32	4 294 967 295 (hex 0xffffffff)	RW	IP Service Activator generates the SOO value itself when required. Valid only if EBGP is the routing protocol. Range: 1–2 147 483 647 plus the default value.
Context	String		RW	
DomainVpnTag	U32	0	RW	To avoid routing loops when you are using OSPF as the routing protocol between a CE and PE device in an MPLS VPN. Routing loops can occur if OSPF routes are passed between PEs in the same network and VPN. The VPN Route tag is one mechanism that can be used to prevent routing loops in multi-homed VPNs.
EigrpAutonomousSystem	U32	0	RW	Specifies the EIGRP ASN for the site.
EigrpSooAsn	U32	0	RW	The Site of Origin will have the form <ASN>:<Origin ID>. EigrpSooAsn is the first part of that value which is <ASN>.
EigrpSooOid	U32	0	RW	The Site of Origin will have the form <ASN>:<Origin ID>. EigrpSooOid is the second part of that value which is <Origin ID>.

Table 4–26 (Cont.) Attributes Common to VirtualCE and VPN Site Objects

Attribute Name	Type	Default	Access	Explanation
EigrpSooRouteMap Name	String		RW	Specify a manually configured Route Map. Make sure that the name contains no spaces; this includes leading or trailing spaces and spaces between characters. If there is any space present in the name, it will not be configured on the device.
UseEigrpSoo	Boolean	False	RW	To enable Site of Origin.
GeneratedSoo	Boolean	False	RW	To have IP Service Activator generate the Site of Origin.
InheritAsn	Boolean	False	RW	To have the site inherit the default EIGRP ASN specified for the VPN.
PublicIpAddress	IPAddress	0.0.0.0	RW	The public IP address of the interface connected to the site.
RedistributeStatic	Boolean	False	RW	Static routes are redistributed into the dynamic routing protocols configured for the site.

Table 4–27 describes attributes specific to VPN Site objects.

Table 4–27 Attributes Specific to VPN Site Objects

Attribute Name	Type	Default	Access	Explanation
BgpAsn	U32	0	RW	BGP Autonomous System Number. Unique number for routing.
RoutingProtocol	Enum	3	RW	Type of routing being used between the PE and CE, relevant to MPLS VPNs only: 0 = EBGP 1 = RIP 3 = None 4 = OSPF 5 = EBGP_OSPF 6 = EBGP_RIP 7 = EIGRP 8 = EBGP_EIGRP
InstallStatic	Boolean	True	RW	True = Static routing is used in conjunction with relevant routing protocol. False = Static routing is not used.
InstallLocalStatic	Boolean	True	RW	True = Static routes defined in the site are not redistributed. False = Static routes defined in the site are redistributed.

Table 4–28 describes attributes specific to Virtual CE Site objects.

Table 4–28 Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
CapabilityVrfLite	Boolean	False	RW	Enables capability vrf-lite for OSPF on VRF-Lite. Value is true or false.
RedistPolicyOspf2Rip	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into RIP.
RedistPolicyEigrp2Rip	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into RIP.
RedistPolicyEbgp2Rip	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into RIP.
RedistPolicyStatic2Rip	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into RIP.
RedistPolicyConnected2Rip	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into RIP.
RedistPolicyOspf2Bgp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into BGP.
RedistPolicyRip2Bgp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into BGP.
RedistPolicyEigrp2Bgp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into BGP.
RedistPolicyStatic2Bgp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into BGP.
RedistPolicyConnected2Bgp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into BGP.
RedistPolicyEigrp2Ospf	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into OSPF.
RedistPolicyRip2Ospf	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into OSPF.

Table 4–28 (Cont.) Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
RedistPolicyEbgp2Ospf	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into OSPF.
RedistPolicyStatic2Ospf	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into OSPF.
RedistPolicyConnected2Ospf	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into OSPF.
RedistPolicyOspf2Eigrp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into EIGRP.
RedistPolicyRip2Eigrp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into EIGRP.
RedistPolicyEbgp2Eigrp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into EIGRP.
RedistPolicyStatic2Eigrp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into EIGRP.
RedistPolicyConnected2Eigrp	String	[no default]	RW	Values for redistribution routemap name (policy): distribution into EIGRP.
RedistBandwidthOspf2Eigrp	U32	0	RW	bandwidth value in Kilobits per second range 0 - 16777214
RedistBandwidthRip2Eigrp	U32	0	RW	bandwidth value in Kilobits per second range 0 - 16777214
RedistBandwidthEbgp2Eigrp	U32	0	RW	bandwidth value in Kilobits per second range 0 - 16777214
RedistBandwidthStatic2Eigrp	U32	0	RW	bandwidth value in Kilobits per second range 0 - 16777214
RedistBandwidthConnected2Eigrp	U32	0	RW	bandwidth value in Kilobits per second range 0 - 4294967295
RedistMetricRip2Ospf	U32	20	RW	Values for redistribution metric: distribution into OSPF. Range is 0 - 16777214

Table 4–28 (Cont.) Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
RedistMetricEigrp2Ospf	U32	20	RW	Values for redistribution metric: distribution into OSPF. Range is 0 - 16777214
RedistMetricEbgp2Ospf	U32	20	RW	Values for redistribution metric: distribution into OSPF. Range is 0 - 16777214
RedistMetricStatic2Ospf	U32	20	RW	Values for redistribution metric: distribution into OSPF. Range is 0 - 16777214
RedistMetricConnected2Ospf	U32	20	RW	Values for redistribution metric: distribution into OSPF. Range is 0 - 16777214
RedistMetricOspf2Rip	U32	2	RW	Values for redistribution metric: distribution into RIP. Range: 0-16
RedistMetricEigrp2Rip	U32	0	RW	Values for redistribution metric: distribution into RIP. Range: 0-16
RedistMetricEbgp2Rip	U32	1	RW	Values for redistribution metric: distribution into RIP. Range: 0-16
RedistMetricStatic2Rip	U32	0	RW	Values for redistribution metric: distribution into RIP. Range: 0-16
RedistMetricConnected2Rip	U32	0	RW	Values for redistribution metric: distribution into RIP. Range: 0-16
RedistMetricOspf2Bgp	U32	0	RW	Values for redistribution metric:distribution into BGP. Range is 0 - 4294967295

Table 4–28 (Cont.) Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
RedistMetricRip2Bgp	U32	1	RW	Values for redistribution metric:distribution into BGP. Range is 0 - 4294967295
RedistMetricEigrp2Bgp	U32	0	RW	Values for redistribution metric:distribution into BGP. Range is 0 - 4294967295
RedistMetricStatic2Bgp	U32	0	RW	Values for redistribution metric:distribution into BGP. Range is 0 - 4294967295
RedistMetricConnected2Bgp	U32	0	RW	Values for redistribution metric:distribution into BGP. Range is 0 - 4294967295
RedistMetricTypeEigrp2Ospf	U32	2	RW	Metric type. Value 1 or 2.
RedistMetricTypeEbgp2Ospf	U32	2	RW	Metric type. Value 1 or 2.
RedistMetricTypeRip2Ospf	U32	2	RW	Metric type. Value 1 or 2.
RedistMetricTypeStatic2Ospf	U32	2	RW	Metric type. Value 1 or 2.
RedistMetricTypeConnected2Ospf	U32	2	RW	Metric type. Value 1 or 2.
RedistributeOspf	Boolean	FALSE	RW	Redistribution of Ospf routes into EBGP. Value is true or false.
RedistributeRip	Boolean	FALSE	RW	Redistribution of Rip routes into EBGP. Value is true or false.
RedistributeEigrp	Boolean	FALSE	RW	Redistribution of Eigrp routes into EBGP. Value is true or false.
RedistributeStatic	Boolean	FALSE	RW	Redistribution of static routes into EBGP. Value is true or false.
RedistributeConnected	Boolean	FALSE	RW	Redistribution of connected routes into EBGP. Value is true or false.
RedistributeEigrpOspf	Boolean	FALSE	RW	Redistribution from Eigrp to Ospf. Value is true or false.

Table 4–28 (Cont.) Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
RedistributeRipOspf	Boolean	FALSE	RW	Redistribution from Rip to Ospf. Value is true or false.
RedistributeEbgpOspf	Boolean	FALSE	RW	Redistribution from Ebgp to Ospf. Value is true or false.
RedistributeStaticOspf	Boolean	FALSE	RW	Redistribution from Static to Ospf. Value is true or false.
RedistributeConnectedOspf	Boolean	FALSE	RW	Redistribution from Connected to Ospf. Value is true or false.
RedistributeEigrpRip	Boolean	FALSE	RW	Redistribution from Eigrp to Rip. Value is true or false.
RedistributeOspfRip	Boolean	FALSE	RW	Redistribution from Ospf to Rip. Value is true or false.
RedistributeEbgpRip	Boolean	FALSE	RW	Redistribution from Ebgp to Rip. Value is true or false.
RedistributeStaticRip	Boolean	FALSE	RW	Redistribution from Static to Rip. Value is true or false.
RedistributeConnectedRip	Boolean	FALSE	RW	Redistribution from Connected to Rip. Value is true or false.
RedistributeOspfEigrp	Boolean	FALSE	RW	Redistribution from Ospf to Eigrp. Value is true or false.
RedistributeRipEigrp	Boolean	FALSE	RW	Redistribution from Rip to Eigrp. Value is true or false.
RedistributeEbgpEigrp	Boolean	FALSE	RW	Redistribution from Ebgp to Eigrp. Value is true or false.
RedistributeStaticEigrp	Boolean	FALSE	RW	Redistribution from Static to Eigrp. Value is true or false.

Table 4–28 (Cont.) Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
RedistributeConnectedEigrp	Boolean	FALSE	RW	Redistribution from Connected to Eigrp. Value is true or false.
OspfDistributeOutAcl	String	[no default]	RW	Named ACL for Distribute In filtering. The access list specified is applied to incoming updates on the selected interface.
VirtualCEModellingOnly	Boolean	True	RW	True = no Virtual CE provisioning is done. False = Virtual CE provisioning is attempted.
RDHighOrder	U32	0	RW	The top 32 bits of the Route Descriptor value.
RDLowOrder	U32	0	RW	The bottom 32 bits of the route Descriptor value.
VrfTableName	String	[no default]	RW	The name of the VRF routing table.
OverrideVrfTable Limit	Boolean	False	RW	True = Use site-specific settings for VRF table limits False = Use domain defaults for VRF table limits.
SaaIpAddr	IPAddress	0.0.0.0	RW	Destination IP address for SAA.
SaaSourceIpAddrMode	Enum	NotConfigured	RW	Valid modes are: 0=NotConfigured, 1=SameAsDestination, 3=SpecifiedAddress. When choosing 3, SaaSourceIpAddr will be used as source IP address for SAA.
SaaSourceIpAddr	IPAddress	0.0.0.0	RW	Source IP Address for SAA.
VrfAwareSla	Boolean	FALSE	RW	Indicates whether to enable the VRF-aware feature using the VirtualCE VRF name.
VrfTableLimit	U32	0	RW	Maximum number of routes allowed in a VRF (0=No limit).

Table 4–28 (Cont.) Attributes Specific to Virtual CE Site Objects

Attribute Name	Type	Default	Access	Explanation
VrfTableLimit Warning	U32	0	RW	Percentage at which to warn of VRF table limits being exceeded. Range: 1–101, where 1–100 = percentage of VrfTableLimit reached warning. 101 = warning when VrfTableLimit reached.
VrfDesc	String	[no default]	RW	Optional string describes the VRF applied to router, if the router supports VRF description string.
EBgpMaxPaths	U32	1	RW	Maximum number of multipaths on EBGP. Range: 1-6. Valid only when protocol is EBGP on Virtual CE.
VrfRedundantPaths	U32	1 (turned off)	RW	The number of device redundant path configurations. Range is 2^{16} plus default value.
RedistributeDefault Route	Boolean	False	RW	True = Redistribute the Default route. False = Don't.
SaaIpAddr	IPAddress	0.0.0.0	RW	Destination IP address for SAA.
SaaSourceIpAddrMode	Enum	NotConfigured	RW	Valid modes are: 0=NotConfigured, 1=SameAsDestination, 3=SpecifiedAddress. When choosing 3, SaaSourceIpAddr will be used as source IP address for SAA.
SaaSourceIpAddr	IPAddress	0.0.0.0	RW	Source IP Address for SAA
VrfAwareSla	Boolean		RW	Indicates whether to enable the VRF-aware feature using the VirtualCE VRF name.

SiteHub Object

A SiteHub object indicates that a site is a hub of a hub and spoke or management VPN. To make an existing site a hub site, create a SiteHub object as a child of the VPN with the same name as the site to be a hub. To make the site a spoke again, delete the SiteHub object.

The SiteHub object has the following object inheritance:

```
SiteHub.Object
```

See Figure 4–16, "Site Object Diagram" for the SiteHub object diagram.

Table 4–29 describes the attribute for the SiteHub object.

Table 4–29 SiteHub Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the site that is a hub in the parent VPN.

SiteL2 Object

SiteL2 objects represent Layer 2 sites. Used when configuring Layer 2 connectivity services.

The SiteL2 object has the following object inheritance:

```
Tls.Object
```

For port-based TLS services (i.e. the ServiceType attribute is PortBased), each Layer 2 site can contain only PE interfaces, and if multiple interfaces are included in the TLS, they must be on the same PE devices. In the case of a port-based TLS, no VLAN will be created on any device and therefore none of the VLAN-related attributes on the SiteL2 objects will be used.

For port and VLAN-based TLS services (i.e. the ServiceType attribute is PortAndVlanBased), Layer 2 sites can contain also a CE/MTU device and both the CE port connecting the CE to the PE and the CE customer facing ports. The CE customer-facing ports must be assigned a role of Local, while the CE-to-PE port must be assigned a role of Access. Both the CE devices and its customer-facing interfaces must be linked to the site if the system is required to configure the CE/MTU. It is recommended that the CE port connecting the CE to the PE is also part of the site. Alternatively, the system will try and find an Access port on the CE device and consider that port as the port connecting the CE to PE.

It is important to note that the same physical interface (both on the PE and on the CE) can be part of multiple L2Site objects. In fact in the context of Layer 2 VPNs, trunk ports cannot be restricted to one customer only. Instead, physical ports can be part of multiple sites as long as the VLAN ID ranges installed on the same CE devices via different TLSs/Sites do not overlap. The EOM will validate that this is the case. It is important to note that this restriction will apply to CE/MTU devices only. In fact it is assumed that PE ports that are part of different PortAndVlan based TLSs.

The EOM also validates that all PE interfaces that are part of a SiteL2 object belong to the same PE device. In fact in the context of a TLS, this is the only meaningful configuration. No peering relationship will be established between PE devices whose interfaces are part of the same site.

The maximum speed specified for inbound and/or outbound rate limiting will be applied to the customer-facing ports in the L2Site. (Customer-facing ports are the PE Access Ports in those sites where there are no MTUs or the CE Local Ports in those sites that contain an MTU.) If this attribute is zero, no rate limiting is applied to the ports. If a port is part of multiple Layer 2 sites with conflicting rate limiting configuration, a warning will be raised.

Figure 4–17 shows the SiteL2 object diagram.

Figure 4–17 SiteL2 Object Diagram

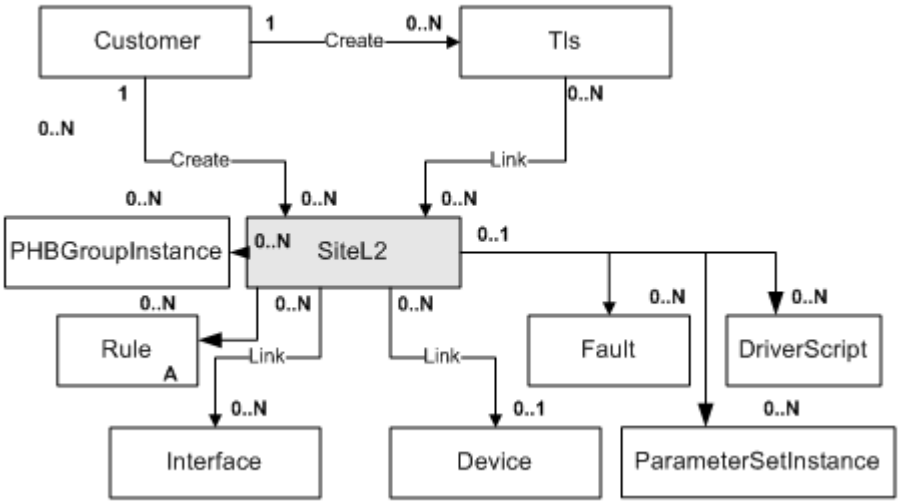


Table 4–30 describes the attributes for the SiteL2 object.

Table 4–30 SiteL2 Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RCW	Name of the Layer 2 site.
Remarks	String	[no default]	RCW	Optional additional comments about this site. Limit of 255 characters.
Contact	String	[no default]	RCW	Name of primary contact at site.
Address	String	[no default]	RCW	Postal address of site.
Telephone	String	[no default]	RCW	Site phone number for contact.
Fax	String	[no default]	RCW	Site fax number for contact.
Email	String	[no default]	RCW	Site e-mail address for contact.
AccountRef	String	[no default]	RCW	Unique account number.
ServiceType	Enum	PortBased	RCW	0 = PortsBased 1 = PortsAndVlanBased

Table 4–31 describes the attributes used if ServiceType = 1.

Table 4–31 SiteL2 Object Attributes Used if ServiceType = 1

Attribute Name	Type	Default	Access	Explanation
SitePortsConfig	Enum	NoEncapsOnlyTaggedFrames	RCW	Configuration of the ports to and from the site. If the site contains only the PE, this configuration will affect the PE access port. If the site includes a CE/MTU device, this will affect the CE customer facing ports. 0 = NoEncapsOnlyTagged Frames. Do not encapsulate and accept tagged frames with VlanIds (trunk port) 1 = EncapsNoTagged Frames. Encapsulate all frames with EncapsVlanId and do not accept previously tagged frames (access port with encapsulation)
VlanIds	String	[no default]	RCW	Lists all the VLAN IDs used in SitePortsConfiguration. List of VLAN IDs to configure on the customer facing interfaces. It is meaningful only if SitePortsConfig = 0.
EncapsVlanId	U32	0	RCW	Specifies the VLAN ID used to encapsulate frames if SitePortsConfig = 1.
VlanType	Enum	VLAN	RCW	

StaticRoute object

A StaticRoute object represents a static route defined for a VPN site (relevant if static routing is used between the PE and CE router). A StaticRoute object can be linked to an interface object that is a child of a site. If the interface is then unlinked from the site, the StaticRoute object will be deleted.

A StaticRoute object can only be created as a child of an interface that is linked to a site, but not linked to an interface.

The StaticRoute object has the following object inheritance:

```
StaticRoute.Object
```

Figure 4–18 shows the StaticRoute object diagram.

Figure 4–18 StaticRoute Object Diagram

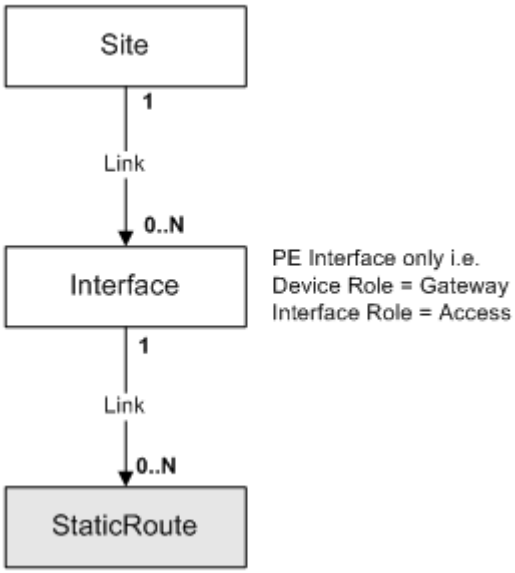


Table 4–32 describes the attributes for the StaticRoute object.

Table 4–32 StaticRoute Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the static route.
IpAddr	IPAddress	0.0.0.0	RW	IP Address to match on.
SubnetMask	IPAddress	0.0.0.0	RW	Subnet mask for route.
NextHop	IPAddress	0.0.0.0	RW	Next hop IP address.
NextHopType	Enum	IpAddressIfName	RW	How the next hop IP address is to be specified in the configuration command. 0=IP Address & I/F 1=IP Address only 2=I/F only 3=Null0 I/F
Metric	U32	1	RW	Metric for the route (that is, a weight).
Global	Boolean	False	RW	True = the given next hop address is in the non-VRF routing table False = the next hop address is not a global route.
Permanent	Boolean	False	RW	True = this route will not be removed, even if the interface shuts down. False = route is not permanent.

Table 4–32 (Cont.) StaticRoute Object Attributes

Attribute Name	Type	Default	Access	Explanation
UseTagNumber	Boolean	False	RW	True = TagNumber will be configured.
TagNumber	U32	0	RW	Label (tag) value that can be used for controlling redistribution of routes through route maps.

Traffic Type Objects

A traffic type is a categorization of specific IP traffic on the network. Traffic types are used as components of rules to define the traffic to which a QoS or security policy is applied.

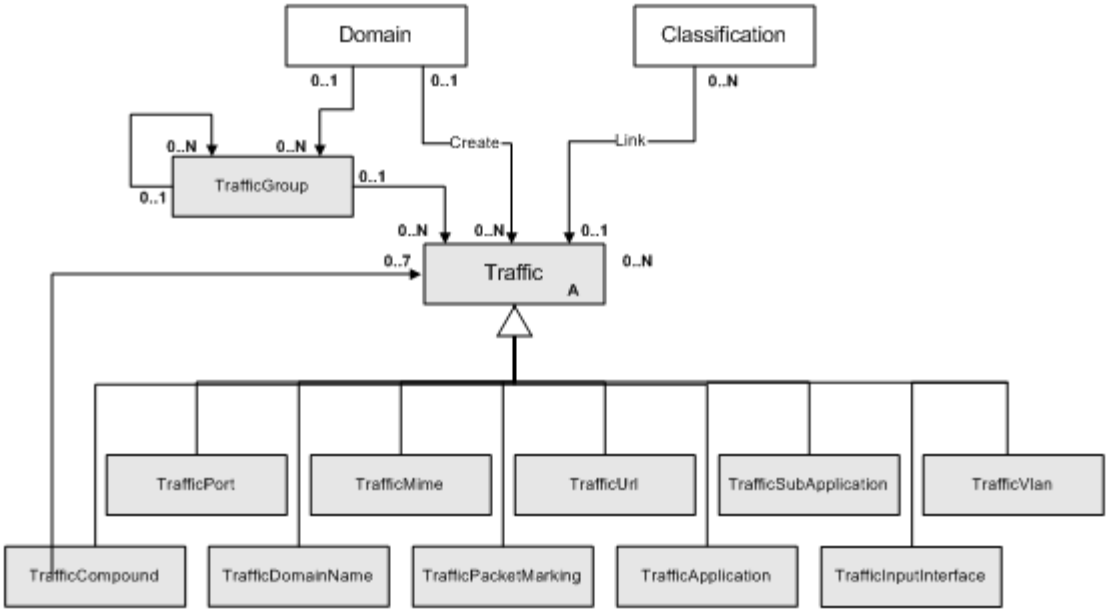
Table 4–33 shows the objects representing different traffic type categorization.

Table 4–33 Traffic Type Object

Object	Purpose
Traffic	Abstract traffic object, for inheritance purposes.
TrafficGroup	Represents a group of traffic types, allowing traffic types to be organized into a hierarchical folder-like structure. Traffic type groups are for administrative purposes only; you cannot apply rules to them.
TrafficCompound	Represents a compound traffic type; a combination of two or more traffic types.
TrafficMime	Represents a MIME-based traffic type, which classifies traffic by MIME type and packet length.
TrafficUrl	Represents a URL-based traffic type, which classifies traffic by URL.
TrafficPort	Represents a port-based traffic type, which classifies traffic by IP port number and IP protocol. Both source and destination ports can be defined either as a single port or a range.
TrafficApplication	Represents an application protocol-based traffic type, which classifies traffic by application protocol name.
TrafficSubApplication	Represents a subapplication-based traffic type, which classifies traffic by subapplication name.
TrafficPacketMarking	Represents a Packet-Marking-based traffic type, which classifies traffic by DiffServ Codepoint/IP Precedence/MPLS Experimental bits/COS/COS Inner.
TrafficDomainName	Represents a domain name-based traffic type, which classifies traffic by DNS Domain Name.
TrafficVlan	Represents traffic based on the Layer 2 VLAN identification number.
TrafficInputInterface	Represents traffic based on the input interface or range of interfaces through which the traffic or IP packet enters

Figure 4–19 shows the traffic type object diagram.

Figure 4–19 Traffic Type Object Diagram



The most common way of classifying traffic is by port number and/or IP protocol, since almost all devices support this. Where supported by devices, it is also possible to identify traffic by packet marking, DNS domain name, application, sub-application, and MIME type or URL for HTTP traffic.

Traffic Object (Abstract)

A Traffic object is an abstract object that defines attributes common to all traffic types.

The abstract traffic object has the following object inheritance:

```
Traffic.Object
```

See Figure 4–19 for the Traffic object diagram.

Table 4–34 describes the attributes for the Traffic object.

Table 4–34 Traffic Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO/RC*	Name of traffic object.
Remarks	String	[no default]	RO/RC*	Optional additional comments. Limit of 255 characters.

* Access is Read Only unless creating a TrafficUrl object, in which case access is Read/Create.

TrafficGroupObject

A TrafficGroup object represents a group [folder] of traffic types. This folder is an administrative entity only, allowing traffic types to be grouped together. These groupings appear in the tree hierarchy but cannot be selected as part of a rule.

The TrafficGroup object has the following object inheritance:

```
TrafficGroup.Traffic.Object
```

See Figure 4–19 for the TrafficGroup object diagram.

The TrafficGroup object has no additional attributes over the Traffic object (abstract).

TrafficCompound Object

A TrafficCompound object allows two or more traffic types to be combined, for example to identify traffic by port/IP protocol and by URL.

The TrafficCompound object has the following object inheritance:

```
TrafficCompound.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficCompound object diagram.

Currently one of each basic type only is allowed. Certain combinations may not in fact make sense, but no restrictions are enforced. The traffic types linked to a TrafficCompound object are not visible via the EOM, but their values are shown as attributes of the TrafficCompound.

The TrafficCompound contains all the attributes of the other Traffic types, as described in Table 4–35.

Table 4–35 TrafficCompound Object Attributes

Attribute Name	Type	Default	Access	Explanation
URL	String		RW	Text string representing a URL.
MIMEType	String	[no default]	RW	Text string representing a MIME type, e.g. text/plain; charset=us-ascii
MinLength	U32	0	RW	Mime: Minimum length of packets in bytes. Range: 0–65535 Must be less than the maximum length.
MaxLength	U32	0	RW	Mime: Maximum length of packets in bytes Range: 0–65535 Must be greater than the minimum length.
DestPortMax	U32	0	RO	Port: Upper bound of destination port range. Same as SourcePortMax to specify a single port. Range: 0–65535
DestPortMin	U32	0	RO	Port: Lower bound of destination port range. Range: 0–65535

Table 4–35 (Cont.) TrafficCompound Object Attributes

Attribute Name	Type	Default	Access	Explanation
IpProtocol	U32	0	RO	Port: Number representing IP protocol (such as 6 for TCP, 17 for UDP, etc). Range: 0–255
SourcePortMax	U32	0	RO	Port: Upper bound of source port range. Same as SourcePortMin to specify a single port. Range: 0–65535
SourcePortMin	U32	0	RO	Port: Lower bound of source port range. Range: 0–65535
ApplicationName	String		RW	Valid name of the application protocol on which the traffic type is based, such as "http".
SubApplicationName	String		RW	Name of subapplication.
DomainName	String		RW	DNS Domain Name (partial match is allowed).
MarkingValue	U32	0	RO	Actual marking value. Ex: For IPPrecedence MarkingType. Range: 0–7
MarkingType	Enum		RO	0 = DSCodepointValue 1 = MplsHeader 2 = FrDe 3 = AtmClp 7 = IPPrecedence 8 = DiscardClass 9 = Trust

TrafficMime Object

A TrafficMime object represents a MIME-based traffic type, which classifies traffic by MIME type and packet length.

The TrafficMime object has the following object inheritance:

```
TrafficMime.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficMime object diagram.

Table 4–36 describes the attributes for the TrafficMime object.

Table 4–36 TrafficMime Object Attributes

Attribute Name	Type	Default	Access	Explanation
MIMEType	String	[no default]	RCW	MIME type.
MinLength	U32	0	RCW	Minimum length in bytes.
MaxLength	U32	0	RCW	Maximum length in bytes.

Table 4–36 (Cont.) TrafficMime Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RCW	Name of the TrafficMime
Remarks	String	[no default]	RCW	Limit of 255 characters.
Id	U32	6331	RO	

TrafficUrlObject

A TrafficUrl object represents a URL-based traffic type, which classifies traffic by URL.

The TrafficUrl object has the following object inheritance:

```
TrafficUrl.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficUrl object diagram.

Table 4–37 describes the attribute for the TrafficUrl object.

Table 4–37 TrafficUrl Object Attributes

Attribute Name	Type	Default	Access	Explanation
URL	String	[no default]	RW	Text string representing a URL. A wildcard (*) at the end of the character string can be used to match multiple URLs.

TrafficPort Object

A TrafficPort object represents a port-based traffic type, which classifies traffic by IP port number and IP protocol.

The TrafficPort object has the following object inheritance:

```
TrafficPort.Traffic.Object
```

Both source and destination ports can be defined either as a single port or a range. To define a single port, set the Min and Max to the same value. If the Min and Max values are both set to 0, no port applies.

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficPort object diagram.

Table 4–38 describes the attributes for the TrafficPort object.

Table 4–38 TrafficPort Object Attributes

Attribute Name	Type	Default	Access	Explanation
SourcePortMin	U32	0	RW	Lower bound of source port range. Range: 0–65535
SourcePortMax	U32	0	RW	Upper bound of source port range. Same as SourcePortMin to specify a single port. Range: 0–65535
DestPortMin	U32	0	RW	Lower bound of destination port range. Range: 0–65535

Table 4–38 (Cont.) TrafficPort Object Attributes

Attribute Name	Type	Default	Access	Explanation
DestPortMax	U32	0	RW	Upper bound of destination port range. Same as SourcePortMax to specify a single port. Range: 0–65535
IpProtocol	U32	0	RW	Number representing IP protocol (such as 6 for TCP, 17 for UDP, etc). Range: 0–255
TcpOptions	U32	0	RW	Bitwise attribute containing various TCP control bits. Do not choose both ACK and RST bits - these two bits comprise the Established status. Urg = 1, Ack = 2, Psh = 4, Rst = 8, Syn = 16, Fin = 32, Established = 128
IcmpOptions	U32	0	RW	Bitwise attribute. Echo-Request = 1, Echo-Reply = 2, TTL-Exceeded = 4, Unreachable = 8, Redirect = 16, Time-Exceeded = 32, Packet-Too-Big = 128, Source-Quench = 256, AdministrativelyProhibited = 512

TrafficApplication Object

A TrafficApplication object represents an application protocol-based traffic type, which classifies traffic by application protocol name.

The TrafficApplication object has the following object inheritance:

```
TrafficApplication.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficApplication object diagram.

Table 4–39 describes the attributes for the TrafficApplication object.

Table 4–39 TrafficApplication Object Attributes

Attribute Name	Type	Default	Access	Explanation
ApplicationName	String		RW	Valid name of the application protocol on which the traffic type is based, such as "http".

Table 4–39 (Cont.) TrafficApplication Object Attributes

Attribute Name	Type	Default	Access	Explanation
Type	Enum	Application	RW	Application = 0, RTP = 1
UDPPortRange	U32	0	RW	Range of the UDP port for RTP. Options are 0-16383
UDPStartPort	U32	2000	RW	UDP start port number for RTP. Range is 2000-65535.

TrafficSubApplication Object

A TrafficSubApplication object represents a subapplication-based traffic type, which classifies traffic by subapplication name.

The TrafficSubApplication object has the following object inheritance:

```
TrafficSubApplication.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficSubApplication object diagram.

Table 4–40 describes the attribute for the TrafficSubApplication object.

Table 4–40 TrafficSubApplication Attributes

Attribute Name	Type	Default	Access	Explanation
ApplicationName	String	[no default]	RO	Name of subapplication.

TrafficPacketMarking Object

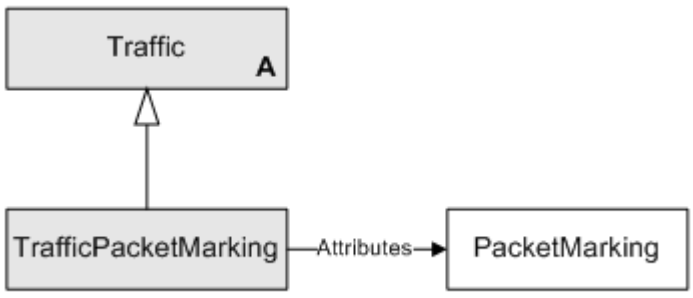
A TrafficPacketMarking object represents a packet marking-based traffic type, which classifies traffic by DiffServ Codepoint, IP Precedence, COS, COS Inner or MPLS Experimental bits.

The TrafficPacketMarking object has the following object inheritance:

```
TrafficPacketMarking.Traffic.Object
```

Figure 4–20 shows the TrafficPacketMarking object diagram.

Figure 4–20 TrafficPacketMarking Object Diagram



The TrafficGroup object has no additional attributes.

TrafficDomainName Object

A TrafficDomainName object represents a domain name-based traffic type, which classifies traffic by DNS Domain Name.

The TrafficDomainName object has the following object inheritance:

```
TrafficDomainName.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficDomainName object diagram.

Table 4–41 describes the attributes for the TrafficDomainName object.

Table 4–41 TrafficDomainName Object Attributes

Attribute Name	Type	Default	Access	Explanation
DomainName	String	[no default]	RO	DNS Domain Name (partial match is allowed).

TrafficVlan Object

A TrafficVlan object represents a VLAN-based traffic type that classifies network traffic on the basis of the Layer 2 VLAN identification number.

The TrafficVlan object has the following object inheritance:

```
TrafficVlan.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficVlan object diagram.

Table 4–42 describes the attributes for the TrafficVlan object.

Table 4–42 TrafficVlan Object Attributes

Attribute Name	Type	Default	Access	Explanation
End	U32	8	RCW	End of the Vlan ID range.
Start	U32	5	RCW	Start of the Vlan ID range.
Name	String	"5-8"	R	Name of the Vlan ID range in the format of "Start-End".
ID	U32	13829	R	

TrafficInputInterface Object

A TrafficInputInterface object represents an input interface-based traffic type. This object classifies IP traffic based on the interface through which the packets enter.

The TrafficInputInterface object has the following object inheritance:

```
TrafficInputInterface.Traffic.Object
```

See Figure 4–19, "Traffic Type Object Diagram" for the TrafficInputInterface object diagram.

Table 4–43 describes the attributes for the TrafficInputInterface object.

Table 4–43 TrafficInputInterface Object Attributes

Attribute Name	Type	Default	Access	Explanation
InterfaceName	String	"Serial1/2"	RCW	Name of the input interface.
Name	String	"InterfaceDemo"	RC	Name of the input interface traffic type.
Remarks	String	[no default]	RC	Limit of 255 characters.
ID	U32	13836	R	

Rule Objects

A QoS or access control policy is implemented by creating and applying a set of rules. Generally, each rule consists of a set of conditions that, when true, result in a set of actions.

There rule objects in the EOM are:

- **RuleAccess**: Represents an access rule, used to deny or permit access to the network for specific identified traffic.
- **RuleClassification**: Represents a classification rule, used to classify, mark, and manage network traffic.
- **RuleGeneric**: Represents a configuration policy, which allows the entry of either a set of raw XML commands, or an HTML-based entry form to collect information which is then converted to XML. The XML commands are passed to the Network Processor.
- **RulePolicing**: Represents a policing rule, used to classify traffic, set bandwidth and burst requirements and define actions for conforming and exceeding traffic.

Rule Object (Abstract)

The abstract Rule object defines attributes common to all rule types.

The abstract Rule object has the following object inheritance:

```
Rule.Object
```

Figure 4–21 shows the Rule object diagram.

Figure 4–21 Rule Object Diagram

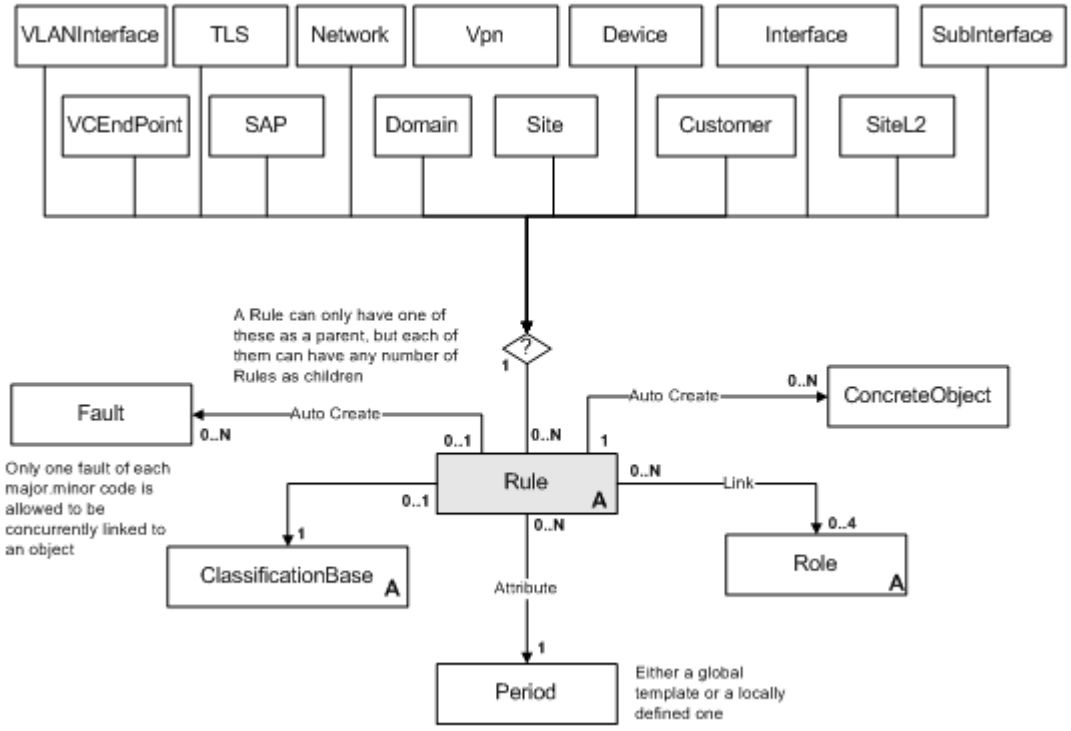


Table 4–44 describes the attributes for the Rule object.

Table 4–44 Rule Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the rule.
Order	U32	0	RW	Rule Order. Indicates pre-requisites ordering for a set of rules. Lowest number (0) indicates highest order. If changed for one rule, all lower priority rules are renumbered.
Disabled	Boolean	False	RO	True = rule is to be disabled at next propagate. False = rule is not to be disabled.
InError	Boolean	False	RO	True = rule is currently failing. False= rule does not contain errors.
Outbound	Boolean	Value depends on rule type	RW	True = rule is active on traffic outbound from an interface. False = rule is not active outbound.
Inbound	Boolean	Value depends on rule type	RW	True = rule is active on traffic inbound to an interface False = rule is not active inbound.

Table 4–44 (Cont.) Rule Object Attributes

Attribute Name	Type	Default	Access	Explanation
StartTime	DateTime	Now	RW	Start date/time - YYYY/MM/DD HH:MM:SS.
EndTime	DateTime	2020/01/01 00:00:00	RW	End date/time - YYYY/MM/DD HH:MM:SS.
DaysOfWeek	U32	127 (i.e. all days selected)	RW	Bitwise value, 1 bit per day of the week: Bit 0 = Monday Bit 1 = Tuesday Bit 2 = Wednesday Bit 3 = Thursday Bit 4 = Friday Bit 5 = Saturday Bit 6 = Sunday

The attribute in Table 4–45 can be used to link directly to other objects in the EOM.

Table 4–45 Rule Object Direct Link Attribute

Attribute Name	Type	Default	Access	Explanation
PeriodName	String	Null	RW	Template name if global, otherwise empty string. Entering a global name links the rule to the Period and sets StartTime, EndTime and DaysOfWeek to Read Only. The global period objects must be linked to the domain to be found.

RuleAccess Object

A RuleAccess object represents an access rule. Access rules are used to provide security by denying or permitting access to the network for specific identified traffic.

The RuleAccess object has the following object inheritance:

RuleAccess.Rule.Object

A RuleAccess object is defined in terms of the following conditions and actions:

Conditions:

- Classifications: One or more Classification objects defining the traffic the rule applies to in terms of source address/mask, destination address/mask and traffic type.
- Date and time: Identified by Period
- Direction: Identified by Outbound and Inbound attributes

Actions:

Permit or deny access to traffic matching the conditions.

Figure 4–22 shows the RuleAccess object diagram.

Figure 4–22 RuleAccess Object Diagram

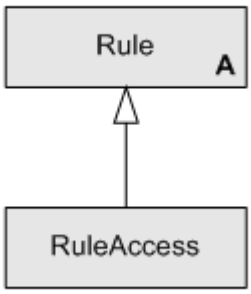


Table 4–46 describes the attributes for the RuleAccess object.

Table 4–46 RuleAccess Object Attributes

Attribute Name	Type	Default	Access	Explanation
Permit	Boolean	True	RCW	Permit or Deny traffic; list is searched until match occurs. True=Explicit permit, stop searching ordered list. False=Deny.
IsNamedAcl	Boolean	False	RCW	Specify if the access rule will be implemented as a named ACL. If set to false, the access rule will be implemented as a numbered ACL.
Ipv4AclName	String	[no default]	RCW	Name of the IPv4ACL implemented for this access rule. If left empty, the system generates the name.
Ipv6AclName	String	[no default]	RCW	Name of the IPv6ACL implemented for this access rule. If left empty, the system generates the name.
AclNumber	U32	168	RCW	Number of the ACL implemented for this access rule. 0 means <Generate> a number.
ManagementOverride	Boolean	False	RCW	To avoid automatic creation of ACL rules. ACL rules are automatically generated to assure SNMP and Telnet access from IP Service Activator to the CE.

Table 4–46 (Cont.) RuleAccess Object Attributes

Attribute Name	Type	Default	Access	Explanation
Log	Boolean	False	RCW	Enables / Disables Cisco ACL logging. Headers of packets affected by the access rule are stored in the routing engine and can be displayed using a show log command. Note: Enabling ACL logging may negatively affect router performance: ■ processor power may be used to perform the logging ■ log files may impact available disk space
Disabled	Boolean	False	RCW	When selected, it switches off the rule.
Fragments	Boolean	False	RCW	When checked, non-initial packet fragments are included for matching. (This relates to the fragments parameter in Cisco ACL statements).
Inbound	Boolean	True	RCW	Applies to inbound traffic.
InError	Boolean	False	RO	True = rule is currently failing. False= rule does not contain errors.
Name	String	"LinkingRuletoNetwork"	RCW	Identifying name of the access rule.
Outbound	Boolean	False	RCW	Applies to outbound traffic.
Order	U32	0	RCW	Rule Order. Indicates pre-requisites ordering for a set of rules. Lowest number (0) indicates highest order. If changed for one rule, all lower priority rules are renumbered.
DaysOfWeek	U32	127	RCW	Bitwise value, 1 bit per day of the week: Bit 0 = Monday Bit 1 = Tuesday Bit 2 = Wednesday Bit 3 = Thursday Bit 4 = Friday Bit 5 = Saturday Bit 6 = Sunday
EndTime	Date	2020/01/01 00:00:00	RCW	End date/time - YYYY/MM/DD HH:MM:SS.

Table 4–46 (Cont.) RuleAccess Object Attributes

Attribute Name	Type	Default	Access	Explanation
PeriodName	String	[no default]	RCW	Template name if global, otherwise empty string. Entering a global name links the rule to the Period and sets StartTime, EndTime and DaysOfWeek to Read Only. The global period objects must be linked to the domain to be found.
StartTime	Date	2008/10/20 05:34:49	RCW	Start date/time - YYYY/MM/DD HH:MM:SS.
ID	U32	6355	RO	Internal ID number of this object; allocated automatically by IP Service Activator.

RuleClassification Object

A RuleClassification object represents a classification rule. Classification rules are used to classify traffic and mark and/or shape traffic.

The RuleClassification object has the following object inheritance:

```
RuleClassification.Rule.Object
```

A RuleClassification object is defined in terms of the following conditions and actions:

Conditions:

- Classifications: One or more Classification objects defining the traffic the rule should apply to in terms of source address/mask, destination address/mask and traffic type.
- Date and time: Identified by Period
- Direction: Identified by Outbound and Inbound attributes

Actions:

- Mark IP packets with DiffServ codepoint, IP Precedence value or MPLS experimental value.
- Apply a maximum bandwidth, if possible (not used at present)
- Apply a guaranteed minimum bandwidth (not used at present).

Figure 4–23 shows the RuleClassification object diagram.

Figure 4–23 RuleClassification Object Diagram

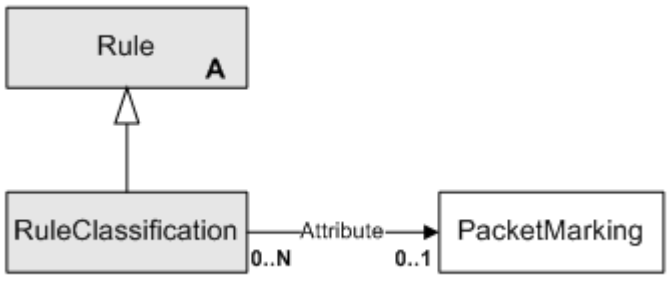


Table 4–47 describes the attributes for the RuleClassification object.

Table 4–47 RuleClassification Object Attributes

Attribute Name	Type	Default	Access	Explanation
BandwidthGuaranteed	U32	0	RCW	Bandwidth level guaranteed for the traffic (in Kbits/s). Must be >= BandwidthLimit, if set. 0 indicates no guarantee.
BandwidthLimit	U32	4294967295	RCW	Maximum bandwidth that can be allocated to traffic (in Kbits/s). Must be <= BandwidthGuaranteed, if set. –1 indicates no limit.
MarkingType	Enum	DSCodepointValue	RO	Marking to be applied 0 = DsCodepoint 1 = MplsExperimental
MarkingValue	U32	0	RO	This value depends on the Marking Type, and defines how to mark matching traffic. Value restricted to 0–255.
Disabled	Boolean	False	RCW	When selected, it switches off the rule.
Inbound	Boolean	False	RCW	Applies to inbound traffic.
InError	Boolean	False	RO	True = rule is currently failing. False= rule does not contain errors.
Name	String	"RuleClassify"	RCW	Name of DiffServ codepoint or MPLS experimental bit with which packets are to be marked. The combo box lists all packet markings that have been set up in IP Service Activator. Where supported, packets can be marked using: ■ IP DiffServ codepoint values ■ MPLS experimental bit values ■ IP Precedence ■ Frame Relay DE bit ■ ATM CLP bit ■ None - no packets are marked
Outbound	Boolean	True	RCW	Applies to outbound traffic.

Table 4–47 (Cont.) RuleClassification Object Attributes

Attribute Name	Type	Default	Access	Explanation
Order	U32	0	RCW	Rule Order. Indicates pre-requisites ordering for a set of rules. Lowest number (0) indicates highest order. If changed for one rule, all lower priority rules are renumbered.
DaysOfWeek	U32	127	RCW	Bitwise value, 1 bit per day of the week: Bit 0 = Monday Bit 1 = Tuesday Bit 2 = Wednesday Bit 3 = Thursday Bit 4 = Friday Bit 5 = Saturday Bit 6 = Sunday
EndTime	Date	2020/01/01 00:00:00	RCW	End date/time - YYYY/MM/DD HH:MM:SS.
PeriodName	String	[no default]	RCW	Template name if global, otherwise empty string. Entering a global name links the rule to the Period and sets StartTime, EndTime and DaysOfWeek to Read Only. The global period objects must be linked to the domain to be found.
StartTime	Date	2008/10/20 05:34:49	RCW	Start date/time - YYYY/MM/DD HH:MM:SS.
Id	U32	6366	RO	Internal ID number of this object; allocated automatically by IP Service Activator.

The attribute in Table 4–48 can be used to link directly to other objects in the EOM

Table 4–48 RuleClassification Object Direct Link Attribute

Attribute Name	Type	Default	Access	Explanation
PacketMarkingName	String	[no default]	RW	Name of the PacketMarking object to use to mark packets.

RuleGeneric Object

A RuleGeneric object represents a configuration policy, which allows the entry of either a set of raw XML commands, or an HTML-based entry form to collect information which is then converted to XML. The XML commands are passed to the Network Processor.

The RuleGeneric object has the following object inheritance:

```
RuleGeneric.Rule.Object
```

You can also specify the ContentValue of a RuleGeneric object as a filename. The filename must be an absolute and complete path ending with ".xml" extension. The file itself can be xml or multi-line. The file contents and namespace are not validated.

Figure 4–24 shows the RuleGeneric object diagram.

Figure 4–24 RuleGeneric Object Diagram

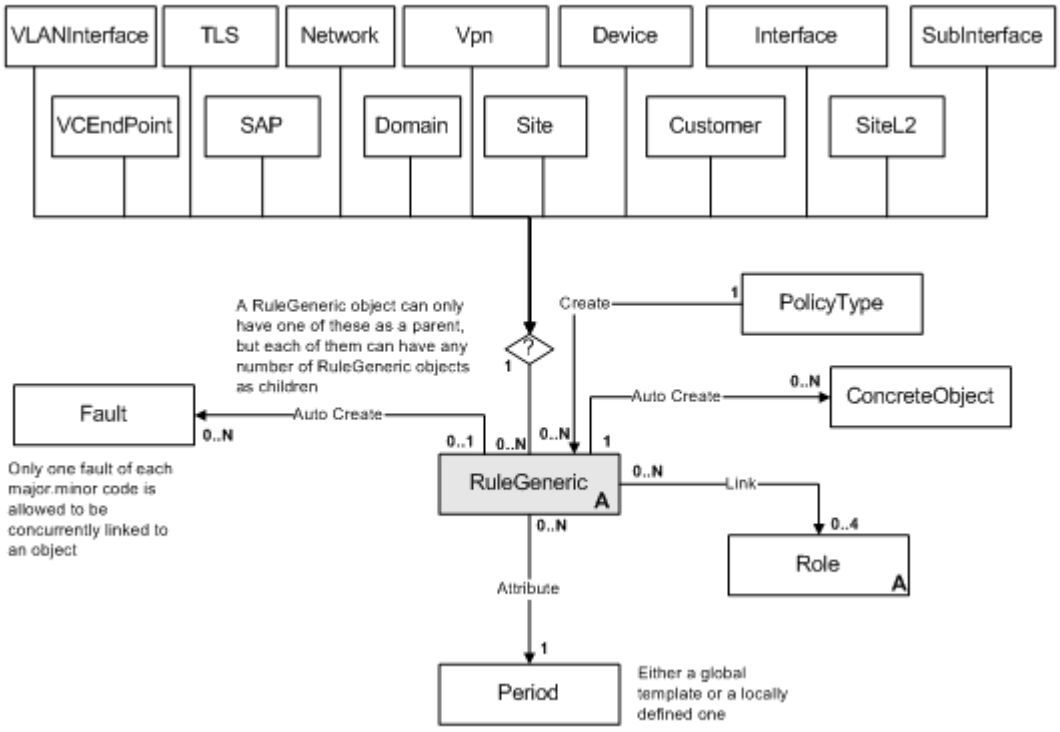


Table 4–49 describes the attributes for the RuleGeneric object.

Table 4–49 RuleGeneric Object Attributes

Attribute Name	Type	Default	Access	Explanation
ContentType	String	[no default]	RCW	Configuration policy type.
ContentValue	String	[no default]	RCW	XML content of the configuration policy.
Disabled	Boolean	False	RCW	If True, no concrete will be created.
Name	String	[no default]	RCW	Name of the configuration policy.
DontCascadeTargets	Boolean	True	RC	Should this policy create concretes at the point of application and on all child Configuration Targets (if set to false) or only on the specific Configuration Target that it is linked to (if set to true)?

Table 4–49 (Cont.) RuleGeneric Object Attributes

Attribute Name	Type	Default	Access	Explanation
Inbound	Boolean	False	RCW	Configuration policy applies to incoming traffic. True = Configuration policy is active on traffic inbound to an interface False = Configuration policy is not active inbound
InError	Boolean	False	RO	True = Configuration policy is currently failing False = Configuration policy does not contain errors
Outbound	Boolean	True	RCW	Configuration policy applies to outbound traffic. True = Configuration policy is active on traffic outbound from an interface False = Configuration policy is not active outbound
Order	U32	0	RCW	Configuration policy Order.
DaysOfWeek	U32	127	RCW	Bitwise value, 1 bit per day of the week: Bit 0 = Monday Bit 1 = Tuesday Bit 2 = Wednesday Bit 3 = Thursday Bit 4 = Friday Bit 5 = Saturday Bit 6 = Sunday
EndTime	Date	2020/01/01 00:00:00	RCW	End date/time - YYYY/MM/DD HH:MM:SS.
PeriodName	String	[no default]	RCW	Template name if global, otherwise empty string. Entering a global name links the rule to the Period and sets StartTime, EndTime and DaysOfWeek to Read Only. The global period objects must be linked to the domain to be found.
StartTime	Date	Now	RCW	Start date/time - YYYY/MM/DD HH:MM:SS.
Id	U32		RO	Internal ID number of this object; allocated automatically by IP Service Activator.

Table 4–49 (Cont.) RuleGeneric Object Attributes

Attribute Name	Type	Default	Access	Explanation
DeviceLevelPolicy	Boolean	False	RCW	True = Configuration policy is applied to a device. False = Configuration policy is applied to an interface. **Note:** If true, on the Role page you can select only a device role and the interface role is not available.

Policy Type Object

The Policy Type object holds information regarding the creation of RuleGeneric objects. Policy Type objects act as templates for Rule Generic Object. They contain the HTML text that is used to populate the forms for a Generic Rule.

Figure 4–25 shows the Policy Type object diagram.

Figure 4–25 Policy Type Object Diagram

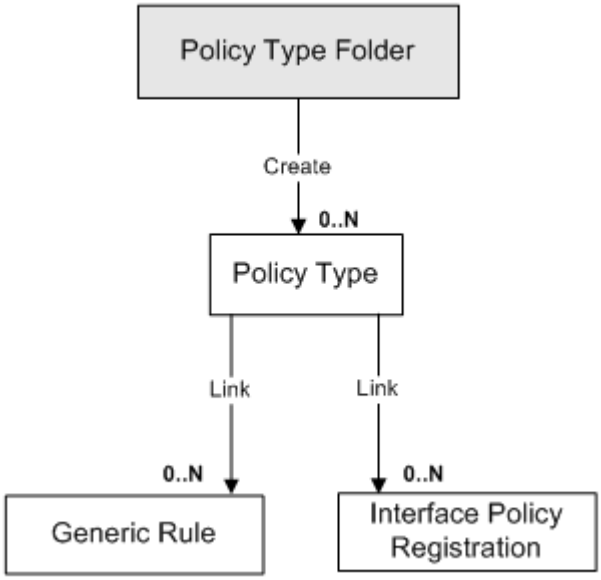


Table 4–50 describes the attributes for the Policy Type object.

Table 4–50 Policy Type Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the policy type.
Category	Enum	Interface, Service, and Unclassified	RW	Configures the category of the policy type.
Type	String	[no default]	RW	Configures the type of policy

Table 4–50 (Cont.) Policy Type Object Attributes

Attribute Name	Type	Default	Access	Explanation
Remarks	String	[no default]	RW	Description of the type of policy. Limit of 255 characters.
HtmlPage	String	[no default]	RW	Imported HTML to be used for data entry.
Schema	String	[no default]	RW	The schema for which the XML is validated.
Disabled	Boolean	False	RW	Specify if the policy type is disabled.
Visible	Boolean	True	RW	Specify if the policy type is visible in the IP Service Activator GUI.

Policy Type Folder Object

The Policy Type Folder object categorizes policy types within the Object Model. A Policy Type must be located in a Policy Type Folder. These folders can also create a hierarchy of Policy Types and Policy Type Folders. Each Policy Type folder can have either a Policy Type Folder or a Policy Type as parent. Only those Policy Types, that have a matching Category Type, can reside under the folder of a particular category.

Table 4–51 describes the attributes for the Policy Type Folder object.

Table 4–51 Policy Type Folder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String		RCW	Name of the configuration policy folder.
Remark	String		RCW	Remarks for this configuration policy folder. Limit of 255 characters.
Category	Enum	Interface, Service, and Unclassified	RCW	Category of the configuration policy folder.

RulePolicing Object

A policing rule is used to police identified traffic between a defined source and destination point. A rule defines the bandwidth and burst requirements for a given traffic type, between a source and destination, and the action to be taken if traffic conforms to or exceeds the requirements.

The RulePolicing object has the following object inheritance:

```
RulePolicing.Rule.Object
```

A Policing Rule is defined in terms of the following conditions and actions:

Conditions

- Classifications: One or more Classification objects defining the traffic the rule should apply to in terms of source address/mask, destination address/mask and traffic type.
- Date and time: Identified by Period

- Direction: Identified by Outbound and Inbound attributes

Actions

- Set CAR parameters: Committed Rate, Normal Burst Size and Excess Burst Size.
- Traffic that conforms to or exceeds a specified bandwidth can be dropped, transmitted, or allowed to continue. In addition, you can choose to re-mark both conforming and exceeding traffic with a different packet marking.

For example,. the following command specifies that conforming traffic is to be transmitted and exceeding traffic is to be re-marked to Bronze:

```
Modify ConformAction="Transmit" ExceedAction="TransmitAndRemark"
ExceedPacketMarking="Bronze"
```

If the actions are Drop or Transmit, then the corresponding packet marking has no effect and may be set to anything (including "").

Figure 4–26 shows the RulePolicing object diagram.

Figure 4–26 RulePolicing Object Diagram

Table 4–52 describes the attributes for the RulePolicing object.

Table 4–52 RulePolicing Object Attributes

Attribute Name	Type	Default	Access	Explanation
CommittedRate	U32	8000	RCW	CAR committed rate (bits per second).
NormalBurstSize	U32	1000	RCW	CAR normal burst size (bytes).
ExcessBurstSize	U32	2000	RCW	CAR excess burst size (bytes).
ExceedAction	Enum	Transmit	RW	Action taken when traffic exceeds specified bandwidth. 1 = Drop 2 = Transmit 10 = TransmitAndRemark 12 = RemarkAndContinue

Table 4–52 (Cont.) RulePolicing Object Attributes

Attribute Name	Type	Default	Access	Explanation
ConformAction	Enum	Transmit	RCW	Action taken when traffic conforms to specified bandwidth. 1 = Drop 2 = Transmit 10 = TransmitAndRemark 12 = RemarkAndContinue
AclNumber	U32	0	RCW	ACL number to be used for CAR policing. 0=Driver chooses. Valid values if user-specified are 100–199 and 2000–2699.
ConformPacketMarking	String	""	RCW	Name of PacketMarking object used to remark conforming packet will be remarked as if ConformAction is TransmitAndRemark or RemarkAndContinue.
ExceedPacketMarking	String	""	RCW	Name of PacketMarking object used to remark exceeding packets if ExceedAction is TransmitAndRemark or RemarkAndContinue.
Disabled	Boolean	False	RCW	The rule is switched off when Disabled is selected.
ExceedAction	Enum	Transmit	RCW	Specifies the action to be taken when traffic exceeds the agreed bandwidth level.
Inbound	Boolean	False	RCW	Checkbox indicating the traffic direction to which the rule applies. It is 'in' in this case.
InError	Boolean	False	RO	
Name	String	"SimpleTest "	RCW	Identifying name of the policing rule.
Outbound	Boolean	True	RCW	Checkbox indicating the traffic direction to which the rule applies. It is 'out' in this case.
Order	U32	0	RCW	
DaysOfWeek	U32	127	RCW	
EndTime	Date	2020/01/01 00:00:00	RCW	
PeriodName	String	""	RCW	
StartTime	Date	2008/10/21 05:59:16	RCW	

Table 4–52 (Cont.) RulePolicing Object Attributes

Attribute Name	Type	Default	Access	Explanation
ID	U32	11503	RO	Internal ID number of this object. It is allocated automatically by the system.

ExceedPacketMarking and ConformPacketMarking are Link by Attribute attributes which take names of other objects.

Period Object

A Period object identifies the particular dates, times and/or days of the week when a rule is to apply. It can be a one-off period or a repeating time period. The period object can either be a globally-named template or a specific local definition of a date/time period.

The Period object has the following object inheritance:

```
Period.Object
```

The Period specifies (possibly contiguous) recurring spans of time of not more than 24 hours. Each span is the same length. The first span starts at the specified **StartTime**. The last span finishes at the specified **EndTime**. After the first span, subsequent spans start exactly 24 hours after the preceding one started. The length of the span is calculated such that the last span will end at the specified **EndTime**. Spans whose start day of the week in UTC corresponds to a zero bit in **DaysOfWeek** are dropped.

Figure 4–27 shows the Period object diagram.

Figure 4–27 Period Object Diagram

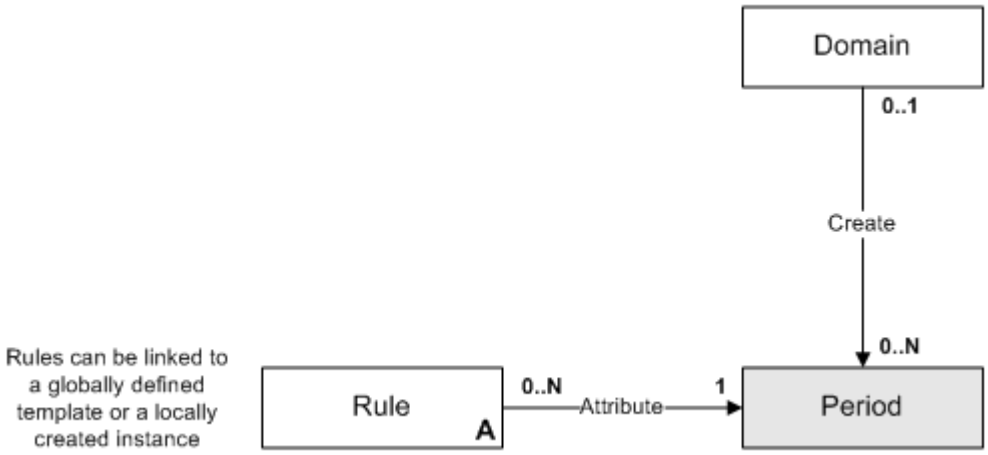


Table 4–53 describes the attributes for the Period object.

Table 4–53 Period Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RO	Template name if global, otherwise NULL.

Table 4–53 (Cont.) Period Object Attributes

Attribute Name	Type	Default	Access	Explanation
StartTime	DateTime	Now	RO	Start date/time - YYYY/MM/DD HH:MM:SS.
EndTime	DateTime	2020/01/01 00:00:00	RO	End date/time - YYYY/MM/DD HH:MM:SS.
DaysOfWeek	U32	127 (i.e. all days selected)	RO	Bitwise value, 1 bit per day: Bit 0 = Monday Bit 1 = Tuesday Bit 2 = Wednesday Bit 3 = Thursday Bit 4 = Friday Bit 5 = Saturday Bit 6 = Sunday

Classification Objects

This section describes the classification objects.

ClassificationBase Object (Abstract)

The ClassificationBase object represents one or many combinations of source addresses and masks, destination addresses and masks and traffic types.

The abstract ClassificationBase object has the following object inheritance:

```
ClassificationBase.Object
```

Figure 4–28 shows the ClassificationBase object diagram.

Figure 4–28 ClassificationBase Object Diagram

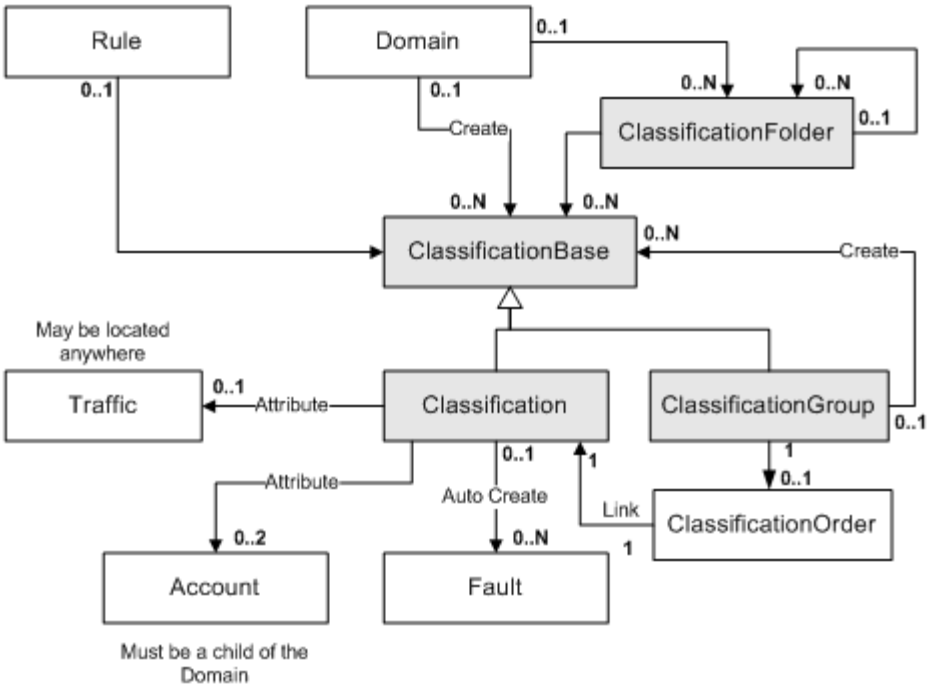


Table 4–54 describes the attributes for the ClassificationBase object.

Table 4–54 ClassificationBase Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	0	RW	Name of the classification object.
Remarks	String	""	RW	Optional additional comments. Limit of 255 characters.

ClassificationGroup Object

The ClassificationGroup object is a composite to allow the aggregation of other Classification objects that are commonly used together. A ClassificationGroup object is a child of either a Domain or a ClassificationFolder object, but not both.

The ClassificationGroup object has the following object inheritance:

```
ClassificationGroup.ClassificationBase.Object
```

See Figure 4–28 for the ClassificationGroup object diagram.

Table 4–55 describes the attributes for the ClassificationGroup object.

Table 4–55 ClassificationGroup Object Attributes

Attribute Name	Type	Default	Access	Explanation
Aggregate	Boolean	True	RW	True if match statements are to be aggregated, otherwise False. Only relevant where the classification is associated with an MQC PHB group via a CoS.
MatchType	Enum	0	RW	Specifies how traffic is matched: 0 = MatchAny 1 = MatchAll Only relevant where the classification is associated with an MQC PHB group via a CoS.
AclIdType	Enum	2	RW	Defines how ACL identifiers are generated: 0 = AutoGenerate 1 = Numbered 2 = Named Only relevant where the classification is associated with an MQC PHB group via a CoS.
AclName	String	Name	RW	Name of the ACL to use. Only relevant where the classification is associated with an MQC PHB group via a CoS.
AclNumber	U32	0	RW	The ACL Number to use. Only relevant where the classification is associated with an MQC PHB group via a CoS.

ClassificationOrder

The ClassificationOrder re-orders the classifications in the group.

The sequence of Classification objects displayed within a Classification Group reflects the order of evaluation. The first Classification in a group matching a packet directs QoS operations on that packet. After the first match, all following Classifications are ignored.

Table 4–56 describes the attributes for ClassificationOrder.

Table 4–56 ClassificationOrder Attributes

Attribute Name	Type	Default	Access	Explanation
orderNumber	U32	0	RW	Relative ordering of a Classification with respect to a parent ClassificationGroup. 0 - 0xFFFFFFFF, default: 0.

Table 4–56 (Cont.) ClassificationOrder Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	ClassificationOrder	RO	Name assigned to the Classification Order.
Id	U32		RO	Unique ID assigned to the Classification Order.

Classification object

Classification Object

The Classification object represents a unique combination of Source Address and Mask, Destination Address and Mask and Traffic Type. A Classification object is a child of either a Domain or a ClassificationFolder object, but not both.

The Classification object has the following object inheritance:

```
Classification.ClassificationBase.Object
```

See Figure 4–28, "ClassificationBase Object Diagram" for the Classification object diagram.

Table 4–57 describes the attributes for the Classification object.

Table 4–57 Classification Object Attributes

Attribute Name	Type	Default	Access	Explanation
SourceIpAddr	IPAddress	0.0.0.0	RW	IPv4 host or network address to identify traffic source. 0.0.0.0/Mask 0 = Any Relevant when AddressType=IPv4
SourceMask	IPAddress	0.0.0.0	RW	Network portion of IPv4 address to match. Relevant when AddressType=IPv4.
DestinationIpAddr	IPAddress	0.0.0.0	RW	IPv4 network or host address to identify traffic destination. 0.0.0.0/Mask 0 = Any Relevant when AddressType=IPv4.
DestinationMask	IPAddress	0.0.0.0	RW	Network portion of IPv4 address to match. Relevant when AddressType=IPv4
Aggregate	Boolean	True	RW	True if match statements are to be aggregated, otherwise False. Only relevant where the classification is associated with an MQC PHB group via a CoS.
MatchType	Enum	0	RW	Specifies how traffic is matched against classifications: 0 = Any 1 = All Only relevant where the classification is associated with an MQC PHB group via a CoS.

Table 4–57 (Cont.) Classification Object Attributes

Attribute Name	Type	Default	Access	Explanation
AclIdType	Enum	2	RW	Defines how ACL identifiers are generated: 0 = AutoGenerate 1 = Numbered 2 = Named Only relevant where the classification is associated with an MQC PHB group via a CoS.
AclName	String	Name	RW	Name of the ACL to use. Only relevant where the classification is associated with an MQC PHB group via a CoS.
AclNumber	U32	0	RW	The ACL Number to use. Only relevant where the classification is associated with an MQC PHB group via a CoS.
AddressType	Enum	IPv4	RW	Selection of address type. Values are: 0 = IPv4 1 = MAC 2 = IPv6
SourceIpv6Addr	String	::/0	RW	IPv6 network or host address to identify traffic source. ::/0 = Any Relevant when AddressType=IPv6
DestinationIpv6Addr	String	::/0	RW	IPv6 network or host address to identify traffic destination. ::/0 = Any Relevant when AddressType=IPv6.
ClassificationMatch	Boolean	TRUE	RW	Whether traffic that matches the specified criteria in the classifier is explicitly included in the processing applied to the classifier or if it is excluded. (corresponds to permit or deny in Cisco ACLs)
SourceMacAddr	String		RW	Identifies the source MAC address by which traffic will be classified.
SourceMacMask	String		RW	Identifies the source MAC mask by which traffic will be classified.
DestinationMacAddr	String		RW	Identifies the destination MAC address by which traffic will be classified.
DestinationMacMask	String		RW	Identifies the destination MAC mask by which traffic will be classified.
Fragments	Boolean	TRUE	RW	When checked, non-initial packet fragments are included for matching. (This relates to the fragments parameter in Cisco ACL statements.)

Table 4–57 (Cont.) Classification Object Attributes

Attribute Name	Type	Default	Access	Explanation
Log	Boolean	TRUE	RW	Enables / Disables Cisco ACL logging. Headers of packets affected by the access rule are stored in the routing engine and can be displayed using a show log command. Note: Enabling ACL logging may negatively affect router performance: · processor power may be used to perform the logging · log files may impact available disk space

The attributes in Table 4–58 can be used to link directly to other objects in the EOM.

Table 4–58 Classification Object Direct Link Attributes

Attribute Name	Type	Default	Access	Explanation
SrcAccountName	String	""	RW	Name of an account to use as source instead of the SourceIpAddress and SourceMask.
DstAccountName	String	""	RW	Name of an account to use as destination instead of the DestinationIpAddr and DestinationMask.
TrafficName	String	""	RW	The name of the traffic object. The traffic object specified may be anywhere in the hierarchy. **Note**: Cannot be a TrafficGroup object.

ClassificationFolder Object

The ClassificationFolder object defines a classification folder, used to contain classification objects, classification group objects, and classification subfolders. A classification folder can be a child of the Domain, or a child of another ClassificationFolder object.

The ClassificationFolderObject has the following object inheritance:

```
ClassificationFolder.ClassificationBase.Object
```

See Figure 4–28, "ClassificationBase Object Diagram" for the ClassificationFolder object diagram.

Table 4–59 describes the attributes for the ClassificationFolder object.

Table 4–59 ClassificationFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the Classification folder.

Table 4–59 (Cont.) ClassificationFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Remarks	String	""	RW	Optional additional comments. Limit of 255 characters.

PHBGroup Objects

This section describes PHBGroup objects.

PHBGroup Object

The PHBGroup object represents a PHB group (an implementation of a specific queuing/shaping mechanism available at an interface).

The PHBGroup object has the following object inheritance:

```
PHBGroup.Object
```

Note: Oracle Communications cannot guarantee compatibility of PHBGroup objects between different releases of IP Service Activator, unlike most other object types. We therefore strongly recommend that any client software you create encapsulates the data manipulated for PHBGroup objects. This ensures a maintainable code base and minimizes the effect of future changes.

The PHBGroup object is a resource-intensive object and we strongly recommend that you create only a limited number. Each PHBGroup object may be used by multiple PHBGroupInstance objects, effectively acting as a template.

Figure 4–29 shows the PHBGroup object diagram.

Figure 4–29 PHBGroup Object Diagram

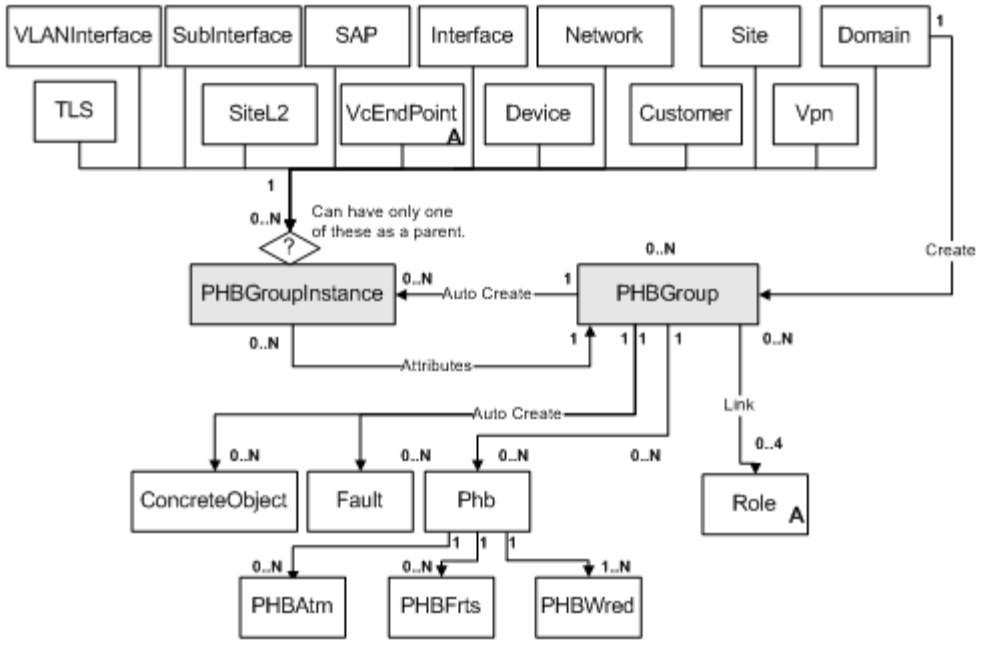


Table 4–60 describes the attributes for the PHBGroup object.

Table 4–60 PHBGroup Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	0	RW	Name of the object.
Description	String		RW	Text entered here is configured as a policy-map description on the device. This behavior is turned off by default. To use this feature, it must be turned on in the capabilities files.
ConfiguredName	String	Value in the **Name** field	RW	Name of the configured PHB Group on the device
Inbound	Boolean	False	RW	True = PHB Group applies to inbound traffic on an interface.
Outbound	Boolean	False	RW	True = PHB group applies to outbound traffic on an interface.
Conflict	Boolean	False	RO	Indicates that this PHB Group is in conflict.
Weight	U32	9	RW	Weight Factor.
ECN	Boolean	False	RW	Modifies WRED performance by marking packets with ECT and ECN bits in the IP header, when congestion occurs in the network.

Table 4–60 (Cont.) PHBGroup Object Attributes

Attribute Name	Type	Default	Access	Explanation
Action	U32	384	RW	Bitwise value indicating the queuing mechanism or traffic shaping mechanism to apply: 1 = PQ 2 = TS 4 = WFQ 8 = WRED 16 = WRR 32 = FRTS 64 = ATMQoS 128 = Default WFQ settings 256 = Default WRED settings Add bit values together to combine queuing mechanisms. For example, a value of **384** indicates Default WFQ (128) and Default WRED (256); a value of **3** indicates PQ (1) and TS (2).
WfqAsPercent	Boolean	False	RW	Use WFQ value as a percentage
ThresholdUnit	Enum	Default	RCW	Unit for Random detect threshold. Valid values are default,bytes and milliseconds.

PHBGroupFolder Object

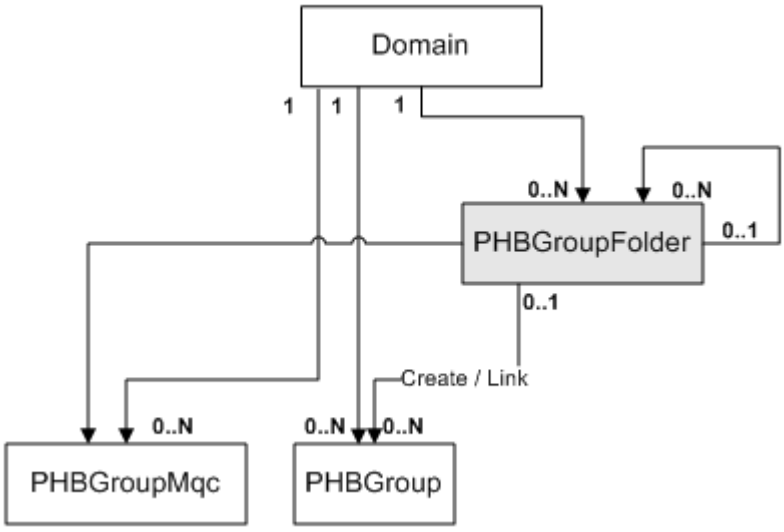
A PHBGroupFolder object represents a folder which contains PHBGroup objects, or other PHBGroupFolder objects, for purposes of organization within the GUI.

The PHBGroupFolder object has the following object inheritance:

```
PHBGroupFolder.Object
```

Figure 4–30 shows the PHBGroupFolder object diagram.

Figure 4–30 PHBGroupFolder Object Diagram



A PHBGroup always has either 1 or 2 parents and is always linked to its parent Domain. It may be linked to zero or one parent PHBGroupFolders. In the IP Service Activator GUI, if a PHBGroup has 2 parents, it will always be displayed under the PHBGroupFolder, not under the PHB Groups folder. If a PHBGroup is created under a folder, it is automatically linked to its parent Domain.

A PHBGroupFolder is a child of either another PHBGroupFolder or the Domain, but not both.

Table 4–61 describes the attributes for the PHBGroupFolder object.

Table 4–61 PHBGroupFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the PHBGroup folder.
Remarks	String	""	RW	Optional additional comments. Limit of 255 characters.

PHBGroupInstance Object

The PHBGroupInstance object represents a particular instance of a PHB group. It includes all the attributes of the domain level PHBGroup template and adds the ability to order them, allowing you to define the precedence in case of multiple PHB groups applied at one level.

The PHBGroupInstance object has the following object inheritance:

```
PHBGroupInstance.PHBGroup.Object
```

A PHBGroupInstance is created by the **use** command. If a PHBGroup is used on one of the PHBGroupInstance possible parents, then a PHBGroupInstance will automatically be created as a child of both the PHBGroup and the other object. For more information, see "The Use Command".

See Figure 4–30, "PHBGroupFolder Object Diagram" for the PHBGroupInstance object diagram.

All the attributes of the corresponding PHBGroup are included in this object (see Table 4–60, " PHBGroup Object Attributes"). In addition, it defines the new attributes described in Table 4–62.

Table 4–62 PHBGroupInstance Object Attributes

Attribute Name	Type	Default	Access	Explanation
Order	U32	0	RW	Order of this PHB group at the local level.
Name	String		RO	The name of the PHB group object that this is an instance of.

PHB Object

The PHB object represents the application of a specific queuing mechanism to a class of service.

Note: The attributes associated with the WFQ queuing mechanism are Read Only in the PHB object. If you wish to configure WFQ, you must use an MQC PHB group. For more information, see "PHBGroupMqc Objects".

See Figure 4–30, "PHBGroupFolder Object Diagram" for the PHB object diagram.

Table 4–63 describes the attributes for the PHB object.

Table 4–63 PHB Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	0	RO	Name of the object. The OIM automatically assigns the value assigned to the object's ClassName attribute.
ID	U32		RO	Unique ID used to reference this object.
Order	U32	0	RCW	Order of this object, compared to other PHB objects.

The attribute in Table 4–64 can be used to link directly to a Cos object in the EOM.

Table 4–64 PHB Object Direct Link Attributes

Attribute Name	Type	Default	Access	Explanation
ClassName	String	""	RCW	Name of the CoS object to match.

Table 4–65 describes the attributes for the WFQ queuing mechanism.

Table 4–65 WFQ Queuing Mechanism Attributes

Attribute Name	Type	Default	Access	Explanation
Weight	U32	100	RCW	WFQ weighting to be allocated to CoS.
QueuePriority	Enum	0	RCW	Priority of this queue. Values: 0 = Low (None) 1 = High (Requested)
DropStrategy	Enum	0	RCW	Values: 0 = None 1 = DefaultWred 2 = WRED 3 = TailDrop
Limit	U32	20	RCW	Packet Limit - set if DropStrategy = TailDrop
SetDe	Boolean	False	RCW	True = set the Frame Relay DE bit.

Table 4–65 (Cont.) WFQ Queuing Mechanism Attributes

Attribute Name	Type	Default	Access	Explanation
WeightFactor	U32	9	RCW	Specifies an exponent weight factor used in calculating the average queue length. Range = 1-16
LimitUnit	Enum	Default	RCW	Unit for QueueLimit. Valid values are default,cells, microseconds,milliseconds,packets and bytes.
WredAggregate	Boolean	False	RCW	True enables Weighted Random Early Detection (WRED) aggregate mode.

Table 4–66 describes the attributes for Priority Queuing.

Table 4–66 Priority Queuing Attributes

Attribute Name	Type	Default	Access	Explanation
Priority	Enum	2	RCW	0 = High 1 = Medium 2 = Normal 3 = Low

Table 4–67 describes the attributes for Rate Limiting.

Table 4–67 Rate Limiting (Traffic Shaping) Attributes

Attribute Name	Type	Default	Access	Explanation
AverageRate	U32	100	RCW	Transmission rate in Kbits/s for the selected class of service.
BurstRate	U32	150	RCW	Transmission burst rate in Kbits/s for the selected class of service.
BurstInterval	U32	10	RCW	Interval, in seconds, over which traffic in the selected class of service is allowed to maintain its maximum burst.

Table 4–68 describes the attributes for Weighted Round Robin (WRR) queuing.

Table 4–68 WRR Queuing Attributes

Attribute Name	Type	Default	Access	Explanation
Bandwidth	U32	2500	RCW	Bandwidth weighting for this class of service.
PacketLimit	U32	20	RCW	Packet Limit.

PHBAtm Object

The PHBAtm object represents the application of an ATM traffic shaping mechanism to a VC endpoint.

See Figure 4–30, "PHBGroupFolder Object Diagram" for the PHBAtm object diagram.

Table 4–69 describes the attributes for the PHBAtm object.

Table 4–69 PHBAtm Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	"PhbAtm"	RO	Identifier used to access the object.
Type	Enum	0	RW	Traffic shaping mechanism to apply: 0 = UBR 1 = CBR 2 = RtVBR 3 = NrtVBR 4 = ABR
PCR	U32	22500	RW	Peak Cell Rate. Used in all service classes. Defined in bits/s. Range 1-45 000 000
SCR	U32	22500	RW	Sustainable Cell Rate. Used for VBR. Defined in bits/s. Range: 1–45 000 000
MBS	U32	32700	RW	Maximum Burst Size. Used for VBR. Specifies the largest burst of data above the ensured rate that will be allowed temporarily on the PVC. Range: 0–65535
MCR	U32	22500	RW	Minimum Cell Rate. Used for ABR. Specifies the minimum value for the ACR. Defined in bits/s. Range: 1–45 000 000.
HoldQueueDepth	U32	0	RW	Maximum number of packets that can be stored in the traffic-shaping queue for an ATM PVC. Range: 5–1024. 0=device default.
TransmitRingLimit	U32	0	RW	Allows the hardware queue depth to be configured. Range: 3–6000. 0=device default.
VcClassName	String		RW	VC class name. The maximum length of the name is 126 characters.
HighWaterMark	U32	0	RCW	1-65535
LowWaterMark	U32	0	RCW	1-200

PHBFrts Object

The PHBFrts object represents the application of a Frame Relay traffic shaping mechanism to an interface.

See Figure 4–30, "PHBGroupFolder Object Diagram" for the PHBFrts object diagram.

Table 4–70 describes the attributes for the PHBFrts object.

Table 4–70 PHBFrts Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	"PhbFrts"	RO	Identifier used to access the object.
Shaping	Boolean	True	RCW	Indicates whether shaping is to be performed.
CIR	String	56000	RCW	Committed Information Rate (bits per second) Range: 1–45 000 000
CIR_In	String	56000	RCW	Committed Information Rate (bits per second) applied as an inbound override for CIR Range: 1–45 000 000
CommittedBurst	String	56000	RCW	Committed burst size (bits). Range: 1000– 160 000 000
CommittedBurst_in	String	56000	RCW	Inbound override for Committed burst size (bits). Range: 1000– 160 000 000
ExcessBurst	String	0	RCW	Excess burst size (bits) Range: 0–16 000 000
ExcessBurst_In	String	0	RCW	Inbound override for excess burst size (bits) Range: 0–16 000 000
MinCIR	String	28000	RCW	Minimum CIR (bits per second). Range: 1000–45 000 000
MinCIR_In	String	28000	RCW	Minimum CIR (bits per second) applied as an inbound override for MinCIR. Range: 1000–45 000 000
BEC[no default]dapt	Boolean	True	RCW	True = adapt to Backward Explicit Congestion Notification flag.
FEC[no default]dapt	Boolean	True	RCW	True = adapt to Forward Explicit Congestion Notification flag.
FRF12Fragment	Boolean	True	RCW	True = Use FRF.12 fragmentation flag.
FRF12Fragment Size	U32	53	RCW	FRF.12 fragment size (bytes). Range: 16–1600
HoldQueueDepth	U32	0	RCW	Maximum number of packets that can be stored in the traffic-shaping queue for a Frame Relay PVC. Range: 1–2048. 0 = device default.
ID	U32	22795	RO	

PHBWred Object

The PHBWred object represents the application of a WRED mechanism to an interface. It is created as a child of a PHB object automatically when the Class of Service that the PHB object represents has Packet Markings attached to it. One PHBWred object is created for each Packet Marking.

A default PHBWred object is created when the 'Default' drop strategy is selected.

See Figure 4–30, "PHBGroupFolder Object Diagram" for the PHBWred object diagram.

Table 4–71 describes the attributes for the PHBWred object.

Table 4–71 PHBWred Object Attributes

Attribute Name	Type	Default	Access	Explanation
MaxDrop	U32	10	RCW	Minimum threshold value for packet discard for selected CoS.
MinDrop	U32	1	RCW	Maximum threshold value for packet discard for selected CoS.
Denominator	U32	10	RCW	Value for Drop probability.

PHBGroupMqc Objects

This section describes the PHBGroupMqc objects.

PHBGroupMqc Object

The PHBGroupMqc object represents an MQC PHB group (the application of a queuing/shaping mechanism via Cisco's Modular QoS CLI).

The PHBGroupMqc object defines generic parameters. The PHBMqc object defines the specific MQC parameters to be applied to a particular CoS handled by an interface.

A PHBGroupMqc object may be linked to multiple PHBMqc objects.

When defining a PHBMqc object to apply congestion avoidance, the drop mechanism may be defined by a standard WRED PHB group.

Figure 4–31 shows the PHBGroupMqc object diagram.

Figure 4–31 PHBGroupMqc Object Diagram

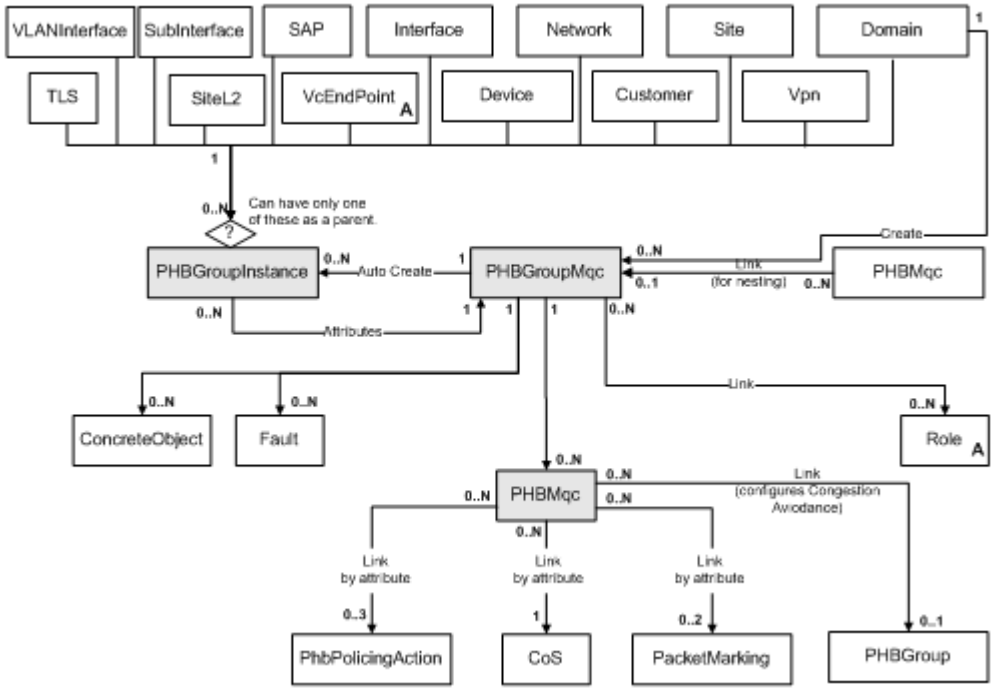


Table 4–72 describes the attributes for the PHBGroupMqc object.

Table 4–72 PHBGroupMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	0	RW	Name of the object.
ConfiguredName	String	Value in the **Name** field	RW	Name of the configured MQC PHB group on the device.
Description	String	--	RW	Text entered here is configured as a policy-map description on the device. This behavior is turned off by default. To use this feature, it must be turned on in the capabilities files.
MaxReservedBandwidth	U32	0	RW	Cisco devices have a default maximum reserve bandwidth value of 75 percent that is designed to leave sufficient bandwidth for overhead traffic. Specify a new value if required.
Inbound	Boolean	False	RW	**True** = Mechanism applies to incoming traffic.
Outbound	Boolean	True	RW	**True** = Mechanism applies to outgoing traffic.

Table 4–72 (Cont.) PHBGroupMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Conflict	Boolean	False	RO	**True** = MQC PHB group is failing. **False** = MQC PHB group is not in error.
LlqBandwidthType	Enum	0	RW	Indicates how LLQ Bandwidth value is interpreted: 0 = Absolute 1 = Percentage 2 = PercentageRemaining 3 = Default 4 = Level
WfqBandwidthType	Enum	0	RW	Indicates how LLQ Bandwidth value is interpreted: 0 = Absolute 1 = Percentage 2 = PercentageRemaining
CreateDefaultPhbMqc	Boolean	True	RC	Applies only when the PhbGroupMqc is created. If set to **True**, IP Service Activator automatically creates a child PhbMqc object named Default Class of Service for the PhbGroupMqc. If set to **True**, you must modify the PhbMqc named Default Class of Service to apply MQC parameters to the default class of service. If set to **False**, you must create one or more PhbMqc child objects for the newly created PhbGroupMqc.

PHBMqc Object

The PHBMqc object represents the application of a specific MQC mechanism to a class of service.

See Figure 4–31, "PHBGroupMqc Object Diagram" for the PHBMqc object diagram.

Table 4–73 describes the attributes for the PHBMqc object.

Table 4–73 PHBMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	0	RO	Name of the object. The OIM automatically assigns the value assigned to the object's CosName attribute.
Order	U32	0	RW	Order of this object, compared to other PHBMqc objects.

Table 4–73 (Cont.) PHBMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
AggregatePolicer	Boolean	FALSE	RW	Set AggregatePolicer to true to activate aggregate policing.
PolicerName	String	[no default]	RW	Specify the aggregate policer name when AggregatePolicer is true.
PoliceRateType	Enum	Absolute	RW	The police rate can be specified in absolute terms (bits/second) or as a percentage of the total available bandwidth. Valid choices are: absolute, percent
VipFairQueue	Boolean	FALSE	RW	When set to true, normal weighted fair queuing is selected for the CoS.
VipFlowQueueLimit	U32	0	RW	Specifies queue limit in number of packets for fair-queuing.
TrustMarking	String	0	RW	This configures the trust state, which selects the value that QoS uses as the source of the internal DSCP value.
CosName	String	0	RW	Name of the Class of Service to be used. Note: This attribute can be used to link directly to a Cos object in the EOM.
Action	U21	0	RW	A bitfield describing the MQC action(s) specified: 0 = None 1 = QueueLLQ 2 = QueueCBWFQ 4 = PoliceSingleRate 8 = PoliceTwoRate 16 = Shape 32 = Mark 64 = Congestion 128 = Nest 256 = RTP Compression 512 = Default WFQ

Table 4–73 (Cont.) PHBMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Queue: LLQWeight	U32	100	RW	Treated as a percentage or an absolute value or a priority level depending on the value of the LlqBandwidthType attribute on the **PhbGroupMqc** object. If expressed as a percentage, the range is 1–100. If expressed as an absolute value, the range is 32–155 000 (Kbits/s) If expressed as a level, the range is 1 - 4.
Queue: LLQBurst	U32	0	RW	Configures the LLQ to accommodate temporary bursts of traffic. Range: 32–2 000 000 bytes. 0 = device default.
Queue: WFQWeight	U32	100	RW	Treated as a percentage or an absolute value depending on the value of the WfqBandwidthType attribute on the **PhbGroupMqc** object. If expressed as a percentage, the range is 1–100. If expressed as an absolute value, the range is 8–2 000 000 (Kbits/s)
Queue: BandwidthOverhead Accounting	Boolean	False	RCW	True indicates that bandwidthoverheadaccounting is enabled.
Queue: BandwidthOffset	U32	2147483647	RCW	Offset value for BandwidthOverheadAccounting. Valid values are from -63 to 63.
Single Rate Policing: SR_CIR	U32	8000	RW	Committed information rate in bits/s. Range: 8000–4 000 000 000
Single Rate Policing: SR_CBS	U32	1000	RW	Committed burst size in bytes. Range: 1000–512 000 000 To load the device default value for **SR_CBS**, send the special value 4 294 967 295.

Table 4–73 (Cont.) PHBMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Single Rate Policing: EBS	U32	1000	RW	Excess burst size in bytes. Range: 1000–512 000 000 Used only if ViolateAction is specified within Policing Action. To load the device default value for **EBS**, send the special value 4 294 967 295.
Two Rage Policing: TR_CIR	U32	8000	RW	Committed information rate in bits/s. Range is 8000-4 000 000 000.
Two Rage Policing: TR_CBS	U32	1000	RW	Committed burst size in bytes. Range is 1000–512 000 000.
Two Rage Policing: PIR	U32	8000	RW	Peak information rate in bits/s. Range is 8000–4 000 000 000.
Two Rage Policing: PBS	U32	8000	RW	Peak burst size in bytes. Range is 1000–512 000 000.
Policing Action: ConformAction	String	[no default]	RW	Name of a **PolicingAction** object used to define the action to perform on conforming traffic.
Policing Action: ExceedAction	String	[no default]	RW	Name of a **PolicingAction** object used to define the action to perform on exceeding traffic.
Policing Action: ViolateAction	String	[no default]	RW	Name of a **PolicingAction** object used to define the action to perform on violating traffic.
Shaping: BasicShaping	Boolean	False	RW	True = When selected, applies basic traffic shaping based on the CIR value only. Applies to Cisco 10000 devices only.
Shaping: Shape_CIR	U32	56000	RW	CIR in bits/s. Range is 8000–154 400 000, must be a multiple of 8000.
Shaping: ShapeCirUnit	Enum	bps	RW	Unit for Shape_CIR. Valid values are bps, kbps, mbps, and gbps.
Shaping: DefaultBc	Boolean	True	RW	True = Use the device default for the Bc value. False = Specify a Bc value.
Shaping: Bc	U32	28000	RW	The normal burst size in bits. Range: 32–154 400 000.
Shaping: DefaultBe	Boolean	True	RW	True = Use the device default for the Be value. False = Specify a Be value.

Table 4–73 (Cont.) PHBMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Shaping: Be	U32	0	RW	The peak burst size in bits. Range: 0–154 400 000
Shaping: ShapeAverage	Boolean	True	RW	True = Use Average rate traffic shaping False = Use Peak rate traffic shaping.
Shaping: ShapingBuffers	U32	0	RW	Number of shaping buffers available to the outgoing queues used for shaping. Range: 1–4096. 0 = device default.
Shaping: FrExtension	Boolean	False	RW	True = Enable the Frame Relay shaping extensions (MinCIR, BEC[no default]dapt, FEC[no default]dapt).
Shaping: MinCIR	U32	56000	RW	Minimum rate to which traffic will be throttled in response to BECN and fecn messages. Range: 8000–154 400 000 The value must be a multiple of 8000. Only valid if FrExtension is True.
Shaping: BEC[no default]dapt	Boolean	False	RW	True = Shaping adapts to BECN. Only valid if FrExtension is true.
Shaping: FEC[no default]dapt	Boolean	False	RW	True = Shaping adapts to FECNs. Only valid if FrExtension is true.
Shaping: ShapeOverheadAccounting	Boolean	False	RCW	True indicates that shapeoverheadaccounting is enabled.
Shaping: ShapeOffset	U32	2147483647	RCW	Offset value for ShapeOverheadAccounting. Valid values are from -63 to 63.
Shaping: WredAggregate	Boolean	False	RCW	True enables Weighted Random Early Detection (WRED) aggregate mode.
Marking: SetDe	Boolean	False	RW	True = set FR DE False = do not set FR DE.
Marking: SetAtmClp	Boolean	False	RW	True = set ATM CLP, False = do not set ATM CLP.
Marking: DscpMarking	String	False	RW	Name of a PacketMarking object representing the DiffServ Codepoint to use.
Marking: IPPrecedenceMarking	String	False	RW	Name of a PacketMarking object representing the IP Precedence to use.

Table 4–73 (Cont.) PHBMqc Object Attributes

Attribute Name	Type	Default	Access	Explanation
Marking: MplsMarking	String	False	RW	Name of a PacketMarking object representing the MPLS Experimental marking to use.
Marking: discardclassmarking	String	False	RW	Name of a PacketMarking object representing the Discard Class to use.
Marking: osMarking	String	[no default]	RW	Name of a packetmarking object representing the COS marking to use.
Marking: CosInnerMarking	String	[no default]	RW	Name of a packetmarking object representing the COS Inner marking to use.
Congestion Avoidance: QueueLimit	U32	0	RW	Limit to apply in packets. The permitted range varies, depending on the device. Check the capabilities. A value of 0 applies the device's default limit.
Congestion Avoidance: WredStrategy	Enum	0	RW	How packets are discarded when congestion occurs 0 = None 1 = Default 2 = WRED. In this case, the **PhbMqc** object must be linked to a **PhbGroup** object.
Classification: ClassMapMatching	Enum	0	R	0 = match-any 1 = match-all
Nesting: NestedPhbGroup	String	[no default]	[no default]	Name of a **PhbGroupMqc** object to be applied to this Class of Service.

PHBPolicingAction Object

The PHBPolicingAction object defines a policing action for use with an MQC PHB group that applies policing. If the object defines a set action using DiffServ codepoints, IP Precedence or MPLS experimental bits, one or two PacketMarking objects must be linked to the PHBPolicingAction object.

The PHBPolicingAction object has the following object inheritance:

```
PHBPolicingAction.Object
```

Figure 4–32 shows the PHBPolicingAction object diagram.

Figure 4–32 PHBPolicingAction Object Diagram

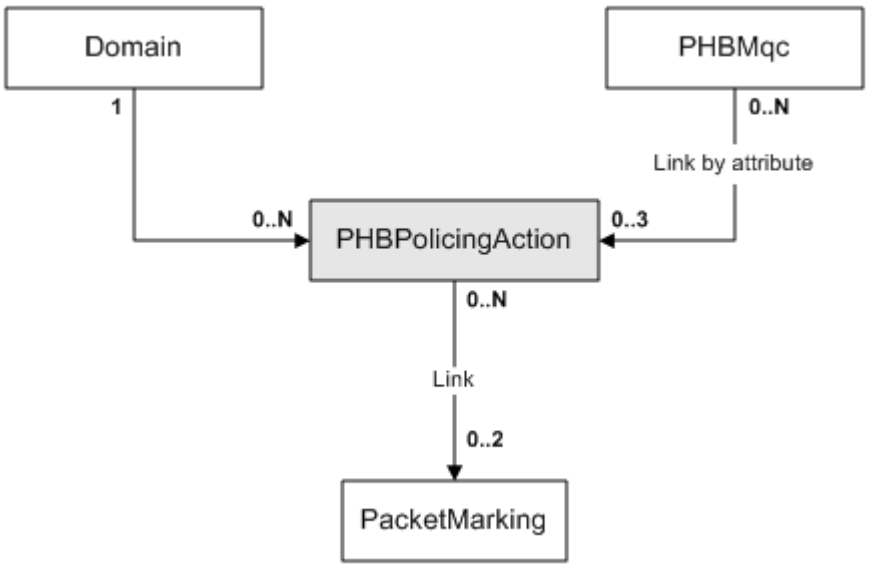


Table 4–74 describes the attributes for the PHBPolicingAction object.

Table 4–74 PHBPolicingAction Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the object.
Type	Enum	0	RW	Enumeration indicating whether to apply a default or user-defined policing action: 0 = User 1 = DefaultConform 2 = DefaultExceed 3 = DefaultViolate
Actions	U32	0	RW	Bitwise value, indicating action(s) to take. 0 = Drop 1 = Transmit 2 = SetClpTransmit 4 = SetFrDeTransmit 8 = SetDscpTransmit 32 = SetIPPrecedenceTransmit 16 = SetMplsExpTransmit 65 = SetPolicedDscpTransmit

Cos Object

The Cos object defines a class of service. The Cos object is a child of the Domain or a COSFolder, but not both.

The Cos object has the following object inheritance:

```
Cos.Object
```

Figure 4–33 shows the Cos object diagram.

Figure 4–33 Cos Object Diagram

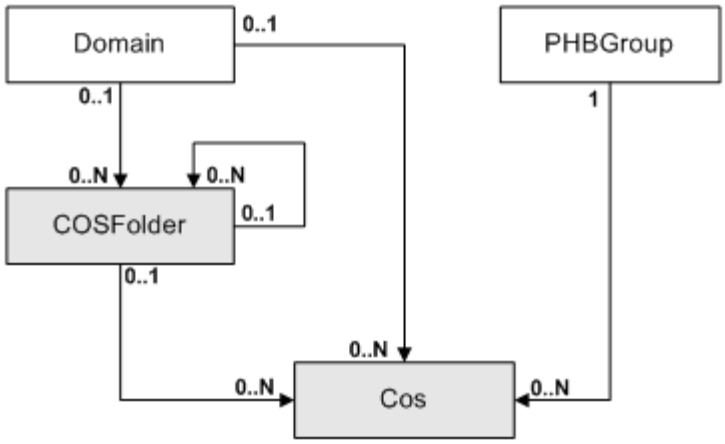


Table 4–75 describes the attributes for the Cos object.

Table 4–75 Cos Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	0	RW	The name of the class of service.
ConfiguredName	String	Value in the **Name** field	RW	Name of the configured Class of Service on the device.
Remarks	String	""	RW	Free-format text field, to add a description of the class of service; limit of 255 characters.
Type	Enum	0	RW	0 = User 1 = Default (IP Class of Service) 2 = NonIP

COSFolder Object

The COSFolder object defines a class of service folder, used to contain class of service (CoS) objects and class of service subfolders. A COSFolder is a child of the Domain or another COSFolder object, but not both.

The COSFolder object has the following object inheritance:

```
COSFolder.Cos.Object
```

See Figure 4–33 for the COSFolder object diagram.

Table 4–76 describes the attributes for the COSFolder object.

Table 4–76 COSFolder Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the class of service folder

Table 4–76 (Cont.) COSFolder Attributes

Attribute Name	Type	Default	Access	Explanation
Remarks	String	""	RW	Optional additional comments. Limit of 255 characters.

Service Group Object

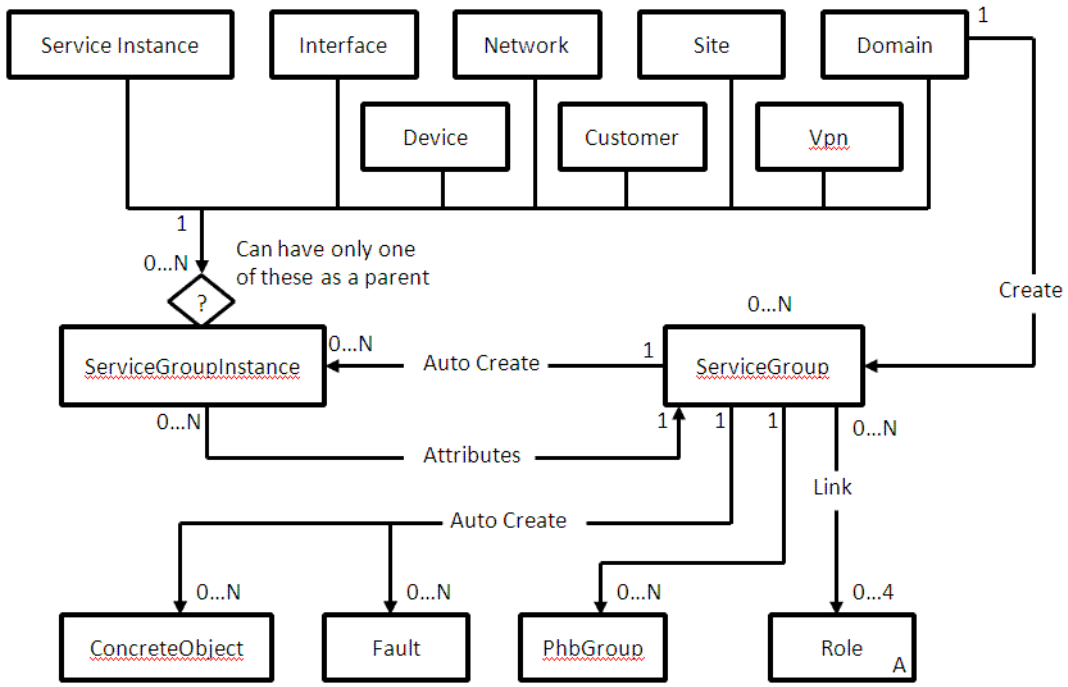
The Service Group feature allows network administrators to create service groups, add PHB groups to those newly created groups, and apply those service groups to service instances.

The Service Group object has the following object inheritance:

```
ServiceGroup.Object
```

Figure 4–34 shows the Service Group object diagram.

Figure 4–34 Service Group Object Diagram



DriverScript Object

Represents a script, written in Python, that when applied to a device, results in the generation of a command script used to configure a device directly.

The DriverScript object has the following object inheritance:

```
DriverScript.Object
```

Figure 4–35 shows the DriverScript object diagram.

Figure 4–35 DriverScript Object Diagram

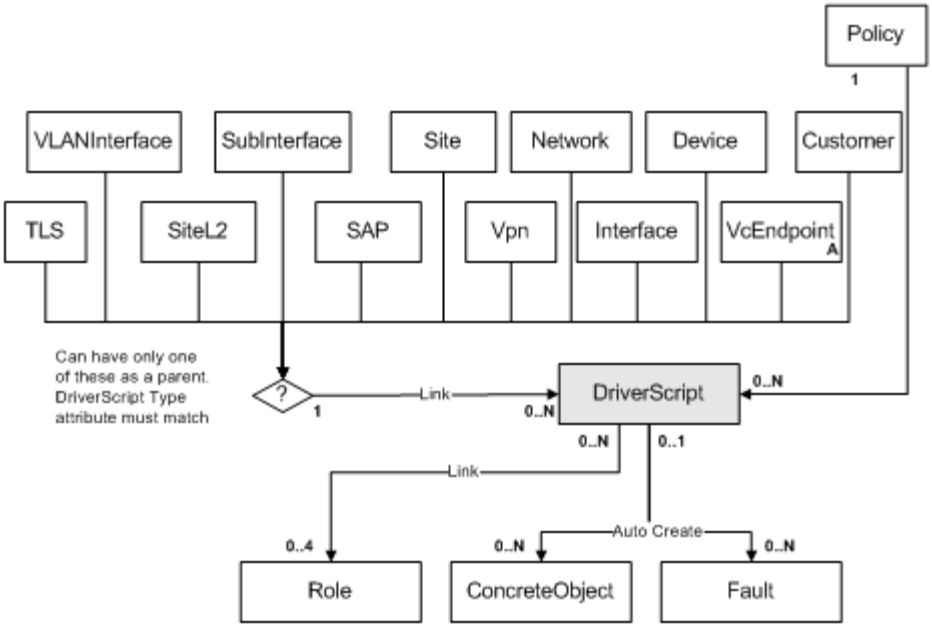


Table 4–77 describes the attributes for the DriverScript object.

Table 4–77 DriverScript Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	Unique	RW	Name of the driver script.
Script	String	""	RW	The text of the Python script. Note that when creating DriverScript objects via the OIM CLI, it is possible to pass the content of a Python script using the following syntax: Script="*pathname*.py" where *pathname* is the full path of the script. Note that no spaces are permitted in *pathname* and the script must have a .py extension. For example: Script="c:\nvram.py"
Type	Enum	0	RW	Type of object this script applies to: 0 = Device 1 = Interface 2 = SubInterface 3 = AtmPvc 4 = FrPvc
DeviceDriverType	String	cisco	RW	Indicates which device driver will run the script.

Table 4–77 (Cont.) DriverScript Object Attributes

Attribute Name	Type	Default	Access	Explanation
When	U32	1	RW	When script is applied: 0 = Before standard configuration changes. 1 = After standard configuration changes.
Repeat	Boolean	False	RW	False = Apply once only. True = Repeat on each propagate.
ReApply	Boolean	False	RW	False = Do not re-apply. True = Repeat on next propagate only. The attribute is automatically set to False after the next propagate.
OnRestart	Boolean	False	RW	False = Don't apply on a device restart. True = Apply on each device restart.
Order	U32	0	RW	Script Order. Lowest number (0) indicates highest priority. Note that if this is changed for one script, all scripts with lower priority numbers are renumbered.
Disabled	Boolean	False	RW	True = script is to be disabled at next propagate. False = script not to be disabled.
InError	Boolean	False	RO	True = script is currently failing. False = script is not in error.
CreateConcretesOnce	Boolean	False	RW	This Boolean is only used for 'Run Once' type scripts. When set to true, it stops the creation of new concretes after the script's first execution.
HasRun	Boolean	False	RO	IP Service Activator sets this value when running the script.

DriverscriptFolder Object

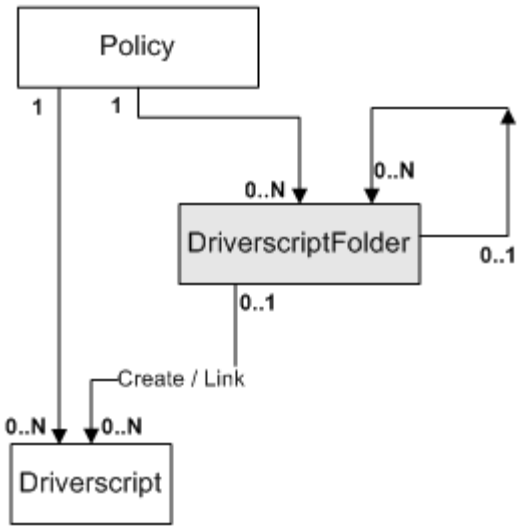
A DriverscriptFolder object represents a folder which contains Driverscript objects, or other DriverscriptFolder objects, for purposes of organization within the GUI.

The DriverscriptFolder object has the following object inheritance:

```
DriverscriptFolder.Object
```

Figure 4–36 shows the DriverscriptFolder object diagram.

Figure 4–36 DriverscriptFolder Object Diagram



A Driverscript always has either 1 or 2 parents and is always linked to its parent Policy. It may be linked to zero or one parent DriverscriptFolders. In the IP Service Activator GUI, if a Driverscript has 2 parents, it will always be displayed under the DriverscriptFolder, not under the Driver Scripts folder. If a Driverscript is created under a folder, it is automatically linked to its parent Policy.

Table 4–78 describes the attributes for the DriverscriptFolder object.

Table 4–78 DriverscriptFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the Driverscript folder.
Remarks	String	""	RW	Optional additional comments. Limit of 255 characters.

Role Objects

This section describes the Role objects.

Role Object (Abstract)

The Role object is used to define the role assigned to a configured object. It is used to determine the targets that policy elements (rules, PHB groups and driver scripts) apply to. Note that rules and PHB groups can have up to four roles (system-defined device role, user-defined device role, system-defined interface role and user-defined interface role).

The abstract Role object has the following object inheritance:

```
Role.Object
```

Figure 4–37 shows the Role object diagram.

Figure 4–37 Role Object Diagram

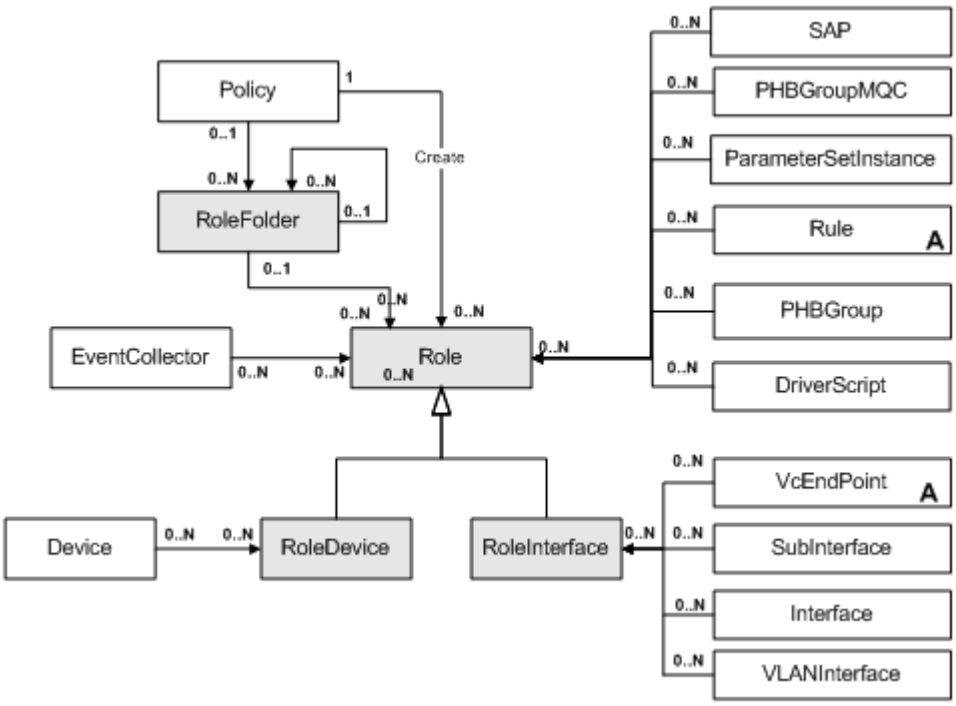


Table 4–79 describes the attributes for the Role object.

Table 4–79 Role Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RC	Name of the role.
Remarks	String	""	RC	Optional additional comments. Limit of 255 characters.

RoleFolder Object

The RoleFolder object defines a role folder, used to contain interface and device role objects and role subfolders. A role folder is the child of the Policy object or another RoleFolder object, but not both.

The RoleFolder object has the following object inheritance:

```
RoleFolder.Role.Object
```

See Figure 4–37 for the RoleFolder object diagram.

Table 4–80 describes the attributes for the RoleFolder object.

Table 4–80 RoleFolder Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the role folder.
Remarks	String	""	RW	Optional additional comments. Limit of 255 characters.

RoleDevice Object

The RoleDevice object is used to define the role assigned to devices. RoleDevice objects can only be linked to Device objects. The RoleDevice object is a child of either the Policy object or a RoleFolder, but not both.

The RoleDevice object has the following object inheritance:

```
RoleDevice.Role.Object
```

See Figure 4–37, "Role Object Diagram" for the RoleDevice object diagram.

Table 4–81 describes the attribute for the RoleDevice object.

Table 4–81 RoleDevice Object Attributes

Attribute Name	Type	Default	Access	Explanation
Type	Enum	0	RO	Defines the type of device role: 0 = UserDefined 1 = Access 2 = Gateway 3 = Core 4 = Any 5 = Shadow

RoleInterface Object

The RoleInterface object is used to define the role assigned to a configured object. RoleInterface objects can be linked to Interface, SubInterface and VcEndPoint objects. The RoleInterface object is a child of either the Policy object or a RoleFolder, but not both.

The RoleInterface object has the following object inheritance:

```
RoleInterface.Role.Object
```

See Figure 4–37 for the RoleInterface object diagram.

Table 4–82 describes the attributes for the RoleInterface object.

Table 4–82 RoleInterface Object Attributes

Attribute Name	Type	Default	Access	Explanation
Type	Enum	0	RO	Defines the type of interface role: 0 = UserDefined 1 = Core 2 = Local 3 = Access 4 = Disabled 5 = Any

PacketMarking Object

The PacketMarking object defines a type of packet marking, which may be a DiffServ codepoint, an IP Precedence value, an MPLS Experimental value, a Frame Relay DE bit setting or an ATM CLP bit setting.

The PacketMarking object has the following object inheritance:

```
PacketMarking.Object
```

Figure 4–38 shows the PacketMarking object diagram.

Figure 4–38 PacketMarking Object Diagram

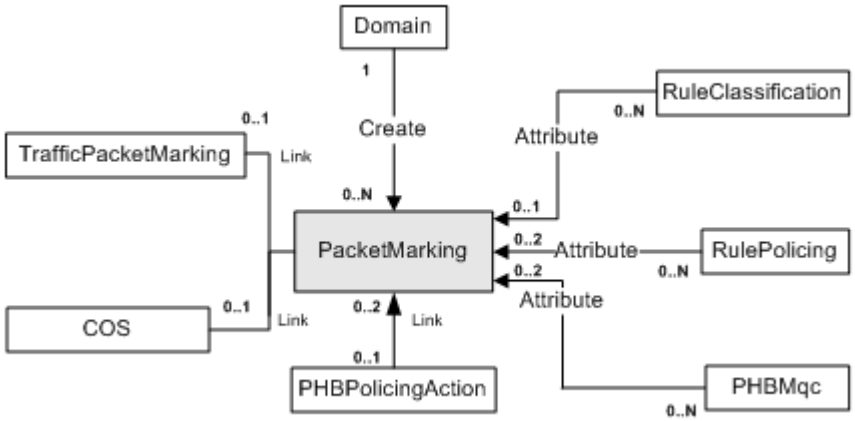


Table 4–83 describes the attributes for the PacketMarking object.

Table 4–83 PacketMarking Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of the object.
MarkingType	Enum	0	RW	0 = DSCodepointValue 1 = MplsHeader 2 = FrDe 3 = AtmClp 7 = IPPrecedence 8 = DiscardClass 9 = Trust 10 = COS 11 = COSInner
MarkingValue	String	0	RW	This value depends on the MarkingType, and defines how to mark traffic. The range is 0–7 for MPLS headers IP Precedence and DiscardClass; 0–63 for DiffServ codepoint. The range is 0-7 for Cos Marking and 0-7 for CosInner marking.
Id	U32	0	RO	Unique ID used to reference this object

ConcreteObject Object

Concrete objects are automatically created and represent the actual application of a Rule, PHB group, VPN or Driver Script to a specific point in the network. Each rule, PHB group, VPN or Driver Script may result in a number of ConcreteObjects.

The ConcreteObject object has the following object inheritance:

```
ConcreteObject.Object
```

Figure 4–39 shows the ConcreteObject object diagram.

Figure 4–39 ConcreteObject Object Diagram

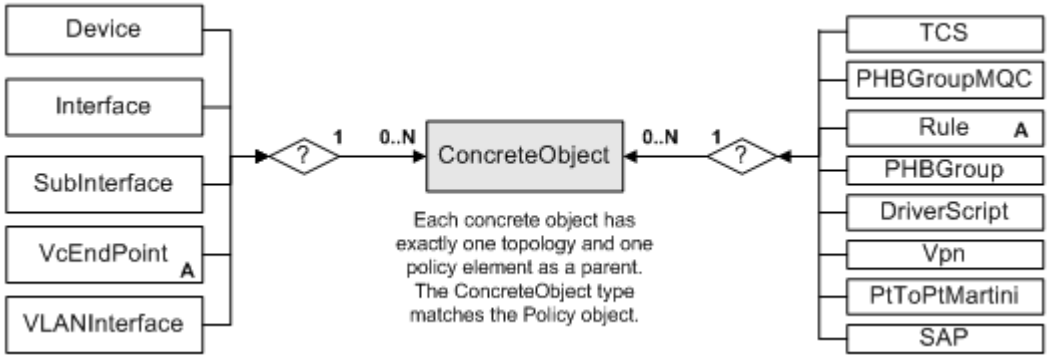


Table 4–84 describes the attributes for the ConcreteObject object.

Table 4–84 ConcreteObject Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RO	Name of the concrete object.
Type	String	"Vpn"	RO	Type of object that created this concrete object. Can be: "RulePolicing" "RuleClassification" "RuleAccess" "DriverScript" "PHBGroup" "Vpn"
Enabled	Boolean	True	RCW	True = policy element is active. False = policy element is disabled.
Conflict	Boolean	False	RO	True = policy element is failing. False = policy element is not in error.
Direction	Enum	Outbound	RO	0 = Inbound 1 = Outbound

Table 4–84 (Cont.) ConcreteObject Object Attributes

Attribute Name	Type	Default	Access	Explanation
State	Enum	0	RO	0 = Inactive 1 = Active 2 = Installed 3 = Failed 4 = Finished 5 = RunFailed 6 = RunOnceFailed 7 = Uninstalled 8 = UninstallFailed Uninstalled indicates successful removal of a policy from the device, after the policy concrete was disabled. UninstallFailed indicates failure to remove a policy from the device, after the policy concrete was disabled. The policy remains on the device, and its concrete remains disabled in the GUI.
VpnOrder	U32	0	RO	When the object is a concrete VPN, this is a unique ID used to reference this object.
AuditMismatchIgnored	Boolean	False	RCW	Set this Boolean so you can ignore an audit mismatch that has been fixed, or was never a real problem. (Set to ignore the mismatch when you do not plan to rerun the device audit again soon.) Unset this Boolean when you no longer wish to ignore an audit mismatch.
AuditState	Enum	NotAvailable	RO	Read-only field displays Passed, Failed, or Not Available
NotificationCount	U32		RO	Concrete state notification counter
ExternalId	U32		RO	External ID of the concrete.

The Topology Model

This section describes the topology model.

Topology Object

The Topology object represents the root of the entire Topology tree.

The Topology object has the following object inheritance:

```
Topology.Object
```

Figure 4–40 shows the Topology object diagram.

Figure 4–40 Topology Object Diagram

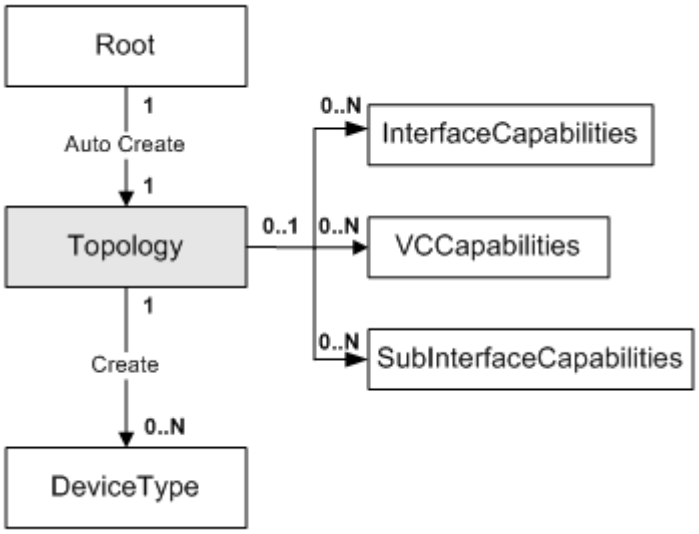


Table 4–85 describes the attributes for the Topology object.

Table 4–85 Topology Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	"Topology"	RO	Always “Topology”.
ReadCommunity	String	public	RW	SNMP community string for Read Access
DiscoveryInProgress	Boolean	False	RO	True when a discovery request is made, set back to false, when request satisfied

Network Object

Each domain has one root-level network object associated with it, which is automatically created with the domain and deleted when the domain is deleted.

The Network object has the following object inheritance:

```
Network.Object
```

A domain’s network can be further partitioned by creating further levels of network objects under the root network object. A network needs to be linked either to a domain or to another network object. Each device within the domain is assigned to one network object.

Figure 4–41 shows the Network object diagram.

Figure 4–41 Network Object Diagram

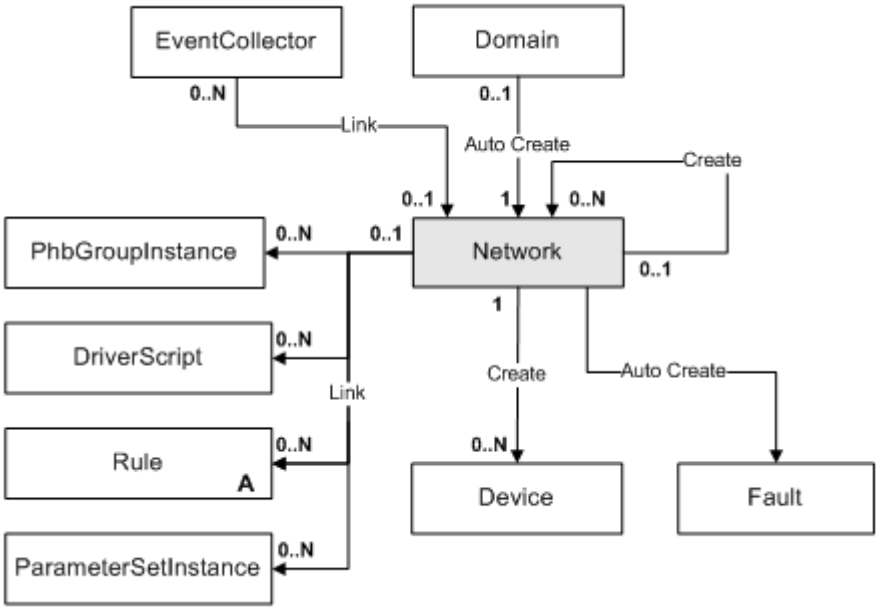


Table 4–86 describes the attributes for the Network object.

Table 4–86 Network Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RO	Name of network. Root network takes the name of the domain by default.
Description	String	""	RW	Free-format comments about the network.
BgpAsn	U32	0	RW	Border Gateway Protocol, Autonomous System Number.
InheritsBgpAsn	Boolean	False	RW	True=Inherits ASN from parent object. False=Does not inherit ASN.
WriteCommunity	Encrypted String	""	RO	The SNMP write community to use when fetching capabilities.
Username	Encrypted String	""	RO	The username for login when fetching capabilities.
Password	Encrypted String	""	RO	The password for login when fetching capabilities.
EnablePassword	Encrypted String	""	RO	The enable password for login when fetching capabilities.
RsaPrivateKey	Encrypted String	""	RO	Name of private key file when fetching capabilities.

Table 4–86 (Cont.) Network Object Attributes

Attribute Name	Type	Default	Access	Explanation
InheritsSecurity	Boolean	False	RO	True = Discovery will use the security attributes of the network object rather than the values entered as parameters to the discover command. False = Discovery does not inherit security settings.
Context	String	""	RW	Local context for driver scripts applied at network level (max 512 bytes).
MaxTransactionSize	U32	0	RW	Maximum number of matches of Configuration Threshold regex pattern allowed in a device configuration session.
IgnoresTransactionSize	Boolean	False	RW	True = Configuration Thresholding is turned on. False = Configuration Thresholding is turned off.
InheritsTransactionSize	Boolean	False	RW	True = all Configuration Thresholding parameters (not just the maximum transaction size) are inherited from parent network. False = parameters are not inherited.
MatchesPatternTransactionSize	String	""	RW	Configuration Threshold regex pattern (limit 127 characters) **Note:** to modify the regular expression, see "Configuration Thresholding Feature: Modifying the Regular Expression"

Device Object

The Device object is used to represent a network node that forwards IP packets, that is, a router or Layer 3 switch, rather than an end system host or server.

The Device object has the following network inheritance:

```
Device.Object
```

Figure 4–42 shows the Device object diagram.

Figure 4–42 Device Object Diagram

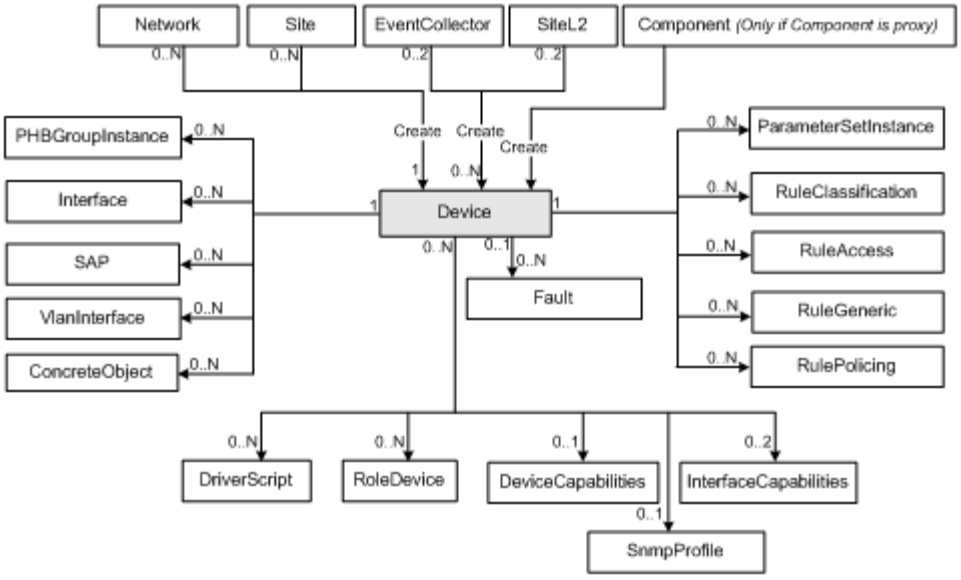


Table 4–87 describes the attributes for the Device object.

Table 4–87 Device Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of device. This is taken from the node name, which is set from the MIB value sysname.0.
Description	String	""	RC	SNMP sysDescription.
Location	String	""	RC	SNMP sysLocation.
IpAddr	IPAddress	0.0.0.0	RW	Address used to communicate with the device.
LoopbackAddr	IPAddress	0.0.0.0	RW	IP address of the loopback interface.
WriteCommunity	Encrypted String	""	RC	SNMP Write community.
ReadCommunity	Encrypted String	""	RC	SNMP Read community.
IsVirtual	Boolean	True	CO	True = Creates a virtual device. Can be set on create and does not appear on the object. False = not a virtual device.
Username	Encrypted String	""	RW	Username.
LoginPassword	Encrypted String	""	WO	Login Password. Required if AccessStyle set to NamedUser, Anonymous, SSH or passwordOnly.
EnablePassword	Encrypted String	""	WO	Enable Password. Required if AccessStyle set to NamedUser, Anonymous, or SSH.
InheritsSecurity	Boolean	True	RW	True = device security options are inherited from the network object. False = security settings are specific for the device.

Table 4–87 (Cont.) Device Object Attributes

Attribute Name	Type	Default	Access	Explanation
State	Enum	7	RO	Last known device state: 0 = Down 1 = Unmanaged 2 = New 3 = NotFound 4 = PreProvisioned 5 = Managed 6 = InterventionRequired 7 = Virtual
UnmanagedAction	Enum	2	RW	The action to take when unmanaging the device and any of its interfaces: 0 = RemoveConfiguration 1 = LeaveConfiguration 2 = UseGlobalSettings
SnmpVn	Enum	3	RW	Version of SNMP to use to interrogate the device: 0 = None 1 = SnmpV1 2 = SnmpV2c 3 = SnmpV1V2c
SnmpRetries	U32	2	RW	The number of retry attempts when sending a PDU (Protocol Data Unit). Range: 0–20
SnmpTimeout	U32	3	RW	Timeout of PDU response Range: 1–30 seconds
SysObjectId	String	""	RC	SNMP obtained unique object identifier.
UpTime	U32	Now	RC	Date and Time since the device is available (seconds).
CommandDeliveryMode	Enum	0	RW	0 = Online: commands delivered to device 1 = OfflineMaintenance: commands not delivered, concretes and other states updated 2 = OfflineTest: commands not delivered
AccessType	Enum	5	RW	0 = NamedUser 1 = Anonymous 2 = TACACS 3 = SNMPv1 4 = SNMPv2c 5 = None 6 = SSH 8 = RSA_SSH 9 = passwordOnly
RsaPrivateKey	Encrypted String	""	RO	Name of SSH RSA private key file (max length 3072 characters).

Table 4–87 (Cont.) Device Object Attributes

Attribute Name	Type	Default	Access	Explanation
Context	String	""	RW	Local context for driver scripts applied to devices (max 512 bytes).
ManualConfig Mode	Enum	3	RW	Action on this device for detecting manual configuration: 0 = Disable 1 = Warn 2 = Fail 3 = Inherit
UseSaaIpAddr	Boolean	False	RW	Indicates if SaaIpAddr is to be used as destination IP address for SAA. If set to false, IpAddr will be used instead.
SaaIpAddr	Boolean	0	RW	Destination IP address for SAA.
SaaSourceIpAddr	Boolean	0	RW	Source IP Address for SAA
SaaSourceIpAddrMode	Enum	NotConfigured	RW	Valid modes are: 0=NotConfigured, 1=SameAsDestination, 2=DeviceManagementAddress, 3=SpecifiedAddress. When choosing 3, SaaSourceIpAddr will be used as source IP address for SAA.
AuditId	U32		RO	
AuditState	Enum	NotAvailable	RO	NotAvailable, Passed, Failed, Error
CliPort	U32	23	RW	CLI/Telnet Port to talk to device (for example, Telnet Port 23).
bgpLocalAs	U32	Set by discovery process	RC	Discovered device ASN. This value overrides the domain ASN for BGP configuration.
EigrpAsn	U32	0	RW	ASN for EIGRP. When specified, this ASN is used to create an EIGRP routing process which will contain the individual vrf-address-family configuration required for each VPN running EIGRP.
LoopBackId	U32	0	RW	The Loopback ID value is used to create a loopback interface name by appending it to the name 'loopback'. For example, if the Loopback ID is 0, the loopback interface name created is 'loopback0'. When a device in this domain is discovered, a check is made to see if a loopback interface matching this text string exists. If it does, the IP address of the loopback interface is stored with the device information. Range: 0–4 294 967 295 Note: Any changes made to the default loopback ID for the domain, or for any devices, have no effect until the affected devices are re-discovered. Note that on Juniper M-series devices, the loopback ID must always be specified as 0 through the user interface.
OverrideLoopbackId	Boolean	False	RW	Set this value to override the default loopback ID specified for the domain for this device.

Table 4–87 (Cont.) Device Object Attributes

Attribute Name	Type	Default	Access	Explanation
DeviceType	String	""	RW	Name of DeviceType object containing details of this type of device. The DeviceType object must be a child of the Topology object. This attribute can be used to link directly to other objects in the EOM.
MaxTransactionSize	U32	0	RW	Maximum number of matches of Configuration Threshold regex pattern allowed in a device configuration session.
IgnoresTransactionSize	Boolean	False	RW	True = Configuration Thresholding is turned on. False = Configuration Thresholding is turned off.
InheritsTransactionSize	Boolean	False	RW	True = all Configuration Thresholding parameters (not just the maximum transaction size) are inherited from parent network. False = parameters are not inherited.
MatchesPatternTransactionSize	String	""	RW	Configuration Threshold regex pattern (limit 127 characters). Note: to modify the regular expression, see "Configuration Thresholding Feature: Modifying the Regular Expression"
StrictClassAggregation	Boolean	False	RW	Enables strict aggregation processing for classifications applied to the current device, which ensures classifications from a contained group are promoted to the parent group, when aggregation is enabled on the contained classification group.
MaxRepetitions	U32	100	RW	Specifies the maximum number of rows that will fetch from a network resource in a single request when SNMP V2c is used for discovery. Permitted range for this filed is 1 to 100. Decrease the value of Max-Repetitions if the routers don't respond to the default value of 100.

DeviceCapabilities Object

The DeviceCapabilities object represents the capabilities and characteristics of a device. One object exists for each device.

Table 4–88 describes the attributes for the DeviceCapabilities object.

Table 4–88 DeviceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
AhMacSecurityAlgorithmsCaps	U32	0	RW	Indicates support for AhMacSecurityAlgorithmsCaps.
CapabilitiesSet	Boolean	False	RW	Indicates support for capabilities sets.
CASupport	Boolean	False	RW	Indicates support for CA.
CompressionAlgorithmCaps	U32	0	RW	Indicates support for CompressionAlgorithmCaps.
DhGroupCaps	U32	0	RW	Indicates support for DhGroupCaps.

Table 4–88 (Cont.) DeviceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
EspSecurityAlgorithmsCaps	U32	0	RW	Indicates support for EspSecurityAlgorithmsCaps.
GreSupport	Boolean	False	RW	Indicates support for Gre.
IkeSecurityAlgorithmsCaps	U32	0	RW	Indicates support for IkeSecurityAlgorithmsCaps.
IPsecModesCaps	U32	0	RW	Indicates support for IPsecModesCaps.
SAANetflowVersionSupport	U32	0	RW	Indicates support for SAANetflowVersion.
SAASupport	Boolean	False	RW	Indicates support for SAA.
SAATypesSupported	U32	0	RW	Indicates support for SAATypes.
SharedKeySupport	Boolean	False	RW	Indicates support for SharedKey.
TlsSupport	Boolean	False	RW	Indicates support for Tls.
TrustedRootCAsSupport	Boolean	False	RW	Indicates support for TrustedRootCAs.
VlansSupport	Boolean	False	RW	Indicates support for Vlans.
Name	String		RO	Name of the object.

DeviceType Object

A DeviceType object represents a device type recognized by IP Service Activator. They are automatically created on startup, and on discovery of a device of a new type.

The DeviceType object has the following object inheritance:

```
DeviceType.Object
```

Figure 4–43 shows the DeviceType object diagram.

Figure 4–43 DeviceType Object Diagram

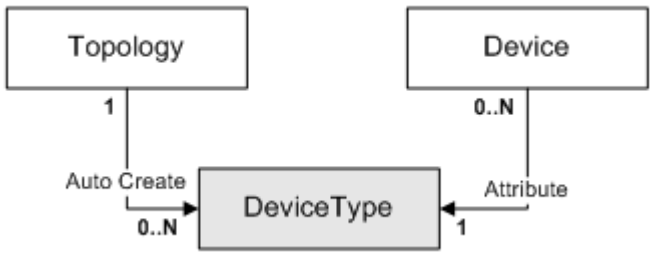


Table 4–89 describes the attributes for the DeviceType object.

Table 4–89 DeviceType Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RW	Name of device type.

Table 4–89 (Cont.) DeviceType Object Attributes

Attribute Name	Type	Default	Access	Explanation
DeviceDriver	String	""	RW	Name of the device driver used to manage this device type.
Vendor	String	""	RW	Company that makes the device (such as Cisco).
Product	String	""	RW	Name of the device (such as 2500).
SoftwareVersion	String	""	RW	Version of software in device (such as 12.1).
SoftwareOs	String	""	RW	Name of operating system (such as IOS).
SysObjectId	String	""	RW	SNMP-obtained unique object identifier.
StrictClassAggregation	Boolean	False	RW	Enables strict classification aggregation on this device

InterfaceCapabilities Object

An InterfaceCapabilities object represents the capabilities and characteristics of an interface on the device. One object exists for each interface. These objects are created by reading the interface table of each device. Capabilities objects are re-used for objects which have the same set of capabilities parameters.

The InterfaceCapabilities object has the following object inheritance:

```
SubInterfaceCapabilities.Object
```

Note: Do not attempt to modify values in the Capabilities object through the OIM. Do not attempt to create Capabilities objects through the OIM. These actions will cause system instability.

Table 4–90 describes the attributes for the InterfaceCapabilities object.

Table 4–90 InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
AccessRulesSupported	U32	0	RCW	Indicates that you can implement access rules on this interface.
ATMQoSSupport	U32	0	RCW	Indicates that you can implement PHB groups using ATM Traffic Shaping on this PVC.
ATMQueueDepthSupport	Boolean	False	RW	Indicates that you can implement PHB groups using ATM queue depth.
ATMTxRingLimitSupport	Boolean	False	RW	Indicates that you can implement PHB groups using ATMTxRingLimit.

Table 4–90 (Cont.) InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
BEC[no default]daptSupport	Boolean	False	RCW	Indicates that you can implement PHB groups using BEC[no default]dapt.
CapabilitiesSet	Boolean	False	RCW	Indicates support for capabilities sets.
CbqClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using Cbq.
CbqClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Cbq.
CbqSupport	U32	0	RCW	Indicates that you can implement PHB groups using Cbq.
ClassificationAccessRules	U32	0	RCW	Indicates that you can implement classification rules on this interface.
ClassificationEnhAccessRules	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface.
ClassificationEnhPolicingRules	U32	0	RW	Indicates that you can implement classification of enhanced policing rules on this interface.
ClassificationEnhServiceRules	U32	0	RW	Indicates that you can implement classification of enhanced service rules on this interface.
ClassificationPolicingRules	U32	0	RCW	Indicates that you can implement classification of policing rules on this interface.
ClassificationServiceRules	U32	0	RCW	Indicates that you can implement classification of service rules on this interface.
DeMarking	Boolean	False	RCW	Indicates support for DeMarking.
FEC[no default]daptSupport	Boolean	False	RCW	Indicates that you can implement PHB groups using FEC[no default]dapt.
FRF12Support	Boolean	False	RCW	Indicates that you can implement PHB groups using FRF12.
FRQueueDepthSupport	Boolean	False	RW	Indicates that you can implement PHB groups using FRQueueDepth.

Table 4–90 (Cont.) InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
FrtsInboundSupport	Boolean	False	RW	Indicates the direction (inbound) in which you can implement PHB groups using FRTS.
FrtsSupport	Boolean	False	RCW	Indicates that you can implement PHB groups using FRTS.
GuaranteesSupported	U32	0	RCW	Indicates that you can implement PHB groups using Guarantees.
IpUnnumberedConfigSupport	Boolean	False	RW	Indicates that you can implement PHB groups using IpUnnumberedConfig.
LabelSwitchingSupport	Boolean	False	RCW	Indicates that you can implement PHB groups using LabelSwitching.
LimitsSupported	U32	0	RCW	Indicates that you can implement PHB groups using Limits.
MarkingPolicingRules	U32	0	RCW	Indicates that you can implement marking policing rules on this interface.
MarkingServiceRules	U32	0	RCW	Indicates that you can implement marking service rules on this interface.
MarkingSupported	U32	0	RCW	Indicates that you can implement PHB groups using Marking.
MqcAggregatePolicer	Boolean	False	RW	Indicates support for MqcAggregatePolicer.
MqcClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using Mqc.
MqcClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Mqc.
MqcLlqBandwidthTypeSupport	U32	0	RCW	Indicates that you can implement PHB groups using MqcLlqBandwidthType.
MqcLlqBurstRateSupport	Boolean	False	RW	Indicates that you can implement PHB groups using MqcLlqBurstRate.
MqcLlqSupport	Boolean	False	RCW	Indicates that this interface supports MqcLlq.

Table 4–90 (Cont.) InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
MqcMarking	U32	0	RCW	Indicates support for MqcMarking.
MqcMaxReservedBandwidth	Boolean	False	RW	Indicates support for MqcMax reserved bandwidth.
MqcNestingSupport	Boolean	False	RCW	Indicates that this interface supports MqcNesting.
MqcPoliceRateTypeSupport	U32	1	RCW	Indicates that this interface supports MqcPoliceRateType.
MqcQueueLimitSupport	Boolean	False	RCW	Indicates that this interface supports MqcQueueLimit.
MqcShapeFrtsSupport	Boolean	False	RCW	Indicates that this interface supports MqcShapeFrts.
MqcShapeSupport	Boolean	False	RCW	Indicates that this interface supports MqcShape.
MqcShapingBuffersSupport	Boolean	False	RW	Indicates that this interface supports MqcShapingBuffers.
MqcSingleRatePoliceAction	U32	0	RCW	Indicates support for MqcSingleRatePoliceAction capability.
MqcSingleRatePoliceSupport	Boolean	False	RCW	Indicates that this interface supports MqcSingleRatePolice.
MqcSupport	U32	0	RCW	Indicates that this interface supports Mqc.
MqcTwoRatePoliceAction	U32	0	RCW	Indicates support for MqcTwoRatePoliceAction capability.
MqcTwoRatePoliceSupport	Boolean	False	RCW	Indicates that this interface supports MqcTwoRatePolice.
MqcWfqBandwidthTypeSupport	U32	0	RCW	Indicates that this interface supports MqcWfqBandwidthType.
MqcWfqSupport	Boolean	False	RCW	Indicates that this interface supports MqcWfq.
MqcWredClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using MqcWred.

Table 4–90 (Cont.) InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
MqcWredClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Mqc.
MqcWredEcn	Boolean	False	RW	Indicates support for MqcWredEcn.
MqcWredSupport	U32	0	RCW	Indicates that this interface supports MqcWred.
Outbound	Boolean	TRUE	RCW	Indicates the direction of the service.
PolicingSupported	U32	0	RCW	Indicates that you can implement policing rules.
PqClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using Pq.
PqClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Pq.
PqSupport	U32	0	RCW	Indicates that this interface supports Pq.
PtToPtEncapsulationCaps	U32	0	RCW	Indicates support for Point to Point EncapsulationCaps capability.
RlimBurst	Boolean	False	RCW	Indicates support for RlimBurst capability.
RlimClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using Rlim.
RlimClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Rlim.
RoutingCaps	bitmap	0	RCW	0 - None 1 - RIP for VPN 2 - Static for VPN 4 - EBGP for VPN 8 - OSPF for VPN 16 - RIP for Virtual CE (not yet supported) 32 - Static for Virtual CE 64 - EBGP for Virtual CE 128 - OSPF for Virtual CE

Table 4–90 (Cont.) InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
VCConfigSupport	Boolean	False	RCW	Indicates that VCConfig is supported.
VpnSupport	bitmap	0	RCW	Indicates that this interface supports VPNs. 0 - None 1 - MPLS 2 - IPsec 4 - Virtual CE
WfqClassify	U32	0	RCW	Indicates that you can implement PHB groups using the WFQ.
WfqClassifyEnh	U32	0	RW	Indicates that you can implement PHB groups using the enhanced WFQ.
WfqHighPriorityWeightAsPercent	Boolean	False	RCW	Indicates that you can implement PHB groups using the WFQ priority as percentage.
WfqLowPriorityWeightAsPercent	Boolean	False	RCW	Indicates that you can implement PHB groups using the WFQ low priority weight as percentage.
WfqPlusWredClassify	U32	0	RCW	Indicates that you can implement PHB groups using the WFQ plus Wred.
WfqPlusWredClassifyEnh	U32	0	RW	Indicates that you can implement PHB groups using the enhanced WFQ plus Wred.
WfqPlusWredSupport	Boolean	False	RCW	Indicates that you can implement PHB groups using the WFQ plus Wred.
WfqSupport	U32	0	RCW	Indicates that you can implement PHB groups using the WFQ.
WredClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using Wred.
WredClassifyEnh	U32	0	RW	Indicates that you can implement classification rules on this interface using enhanced Wred.
WredEcn	Boolean	False	RW	Indicates support for WredEcn.

Table 4–90 (Cont.) InterfaceCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
WredSupport	U32	0	RCW	Indicates that you can implement PHB groups using the WRED.
WrrClassify	U32	0	RCW	Indicates that you can implement classification rules on this interface using Wrr.
WrrClassifyEnh	U32	0	RW	Indicates that you can implement classification rules on this interface using enhanced Wrr.
WrrSupport	U32	0	RCW	Indicates that you can implement PHB groups using the WRR.
Name	String	"Interface Capabilities"	R	Name of the object.
Id	U32	490	R	The unique ID used to reference this object.

VCCapabilities Object

A VCCapabilities object represents the capabilities and characteristics of a virtual circuit on the device. One object exists for each VC. Capabilities objects are re-used for objects which have the same set of capabilities parameters.

The VCCapabilities object has the following object inheritance:

```
VCCapabilities.InterfaceCapabilities.Object
```

See Figure 4–44, "Interface Object Diagram" for the VCCapabilities object diagram.

Table 4–91 describes the attributes for the VCCapabilities object.

Table 4–91 VCCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
CbqClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Cbq.
ClassificationEnhPolicingRules	U32	0	RW	Indicates that you can implement classification using enhanced policing rules.
ClassificationEnhServiceRules	U32	0	RW	Indicates that you can implement classification using enhanced service rules.

Table 4–91 (Cont.) VCCapabilities Object Attributes

Attribute Name	Type	Default	Access	Explanation
MqcClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Mqc.
MqcWredClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Mcq and Wred.
PqClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Pq.
RlimClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Rlim.
WfqClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Wfq.
WfqPlusWredClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Wfq and Wred.
WredClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Wred.
WrrClassifyEnh	U32	0	RW	Indicates that you can implement enhanced classification rules on this interface using Wrr.

Interface Object

An Interface object represents an interface on the device. One object exists for each interface. These objects are created by reading the interface table of each device.

The Interface object has the following object inheritance:

```
Interface.Object
```

Some of the attributes on an Interface object are only meaningful when the Interface is linked as a child of a site object. The attributes cannot be modified when not linked to a site, and if the interface is unlinked from the site then the attributes will automatically be set to null values. The affected attributes are marked 'Site Only' in Table 4–92.

Only attributes related to the selected routing protocol are displayed as part of the Interface attributes. For example, if the routing protocol for the site is set to EBGP, then

only the EBGP-specific routing protocol attributes are displayed as part of the Interface attributes.

If the routing protocols set for the site include EBGP and if the site has more than one neighbor, the attributes for the first neighbor are displayed as part of the Interface attributes.

Figure 4–44 shows the Interface object diagram.

Figure 4–44 Interface Object Diagram

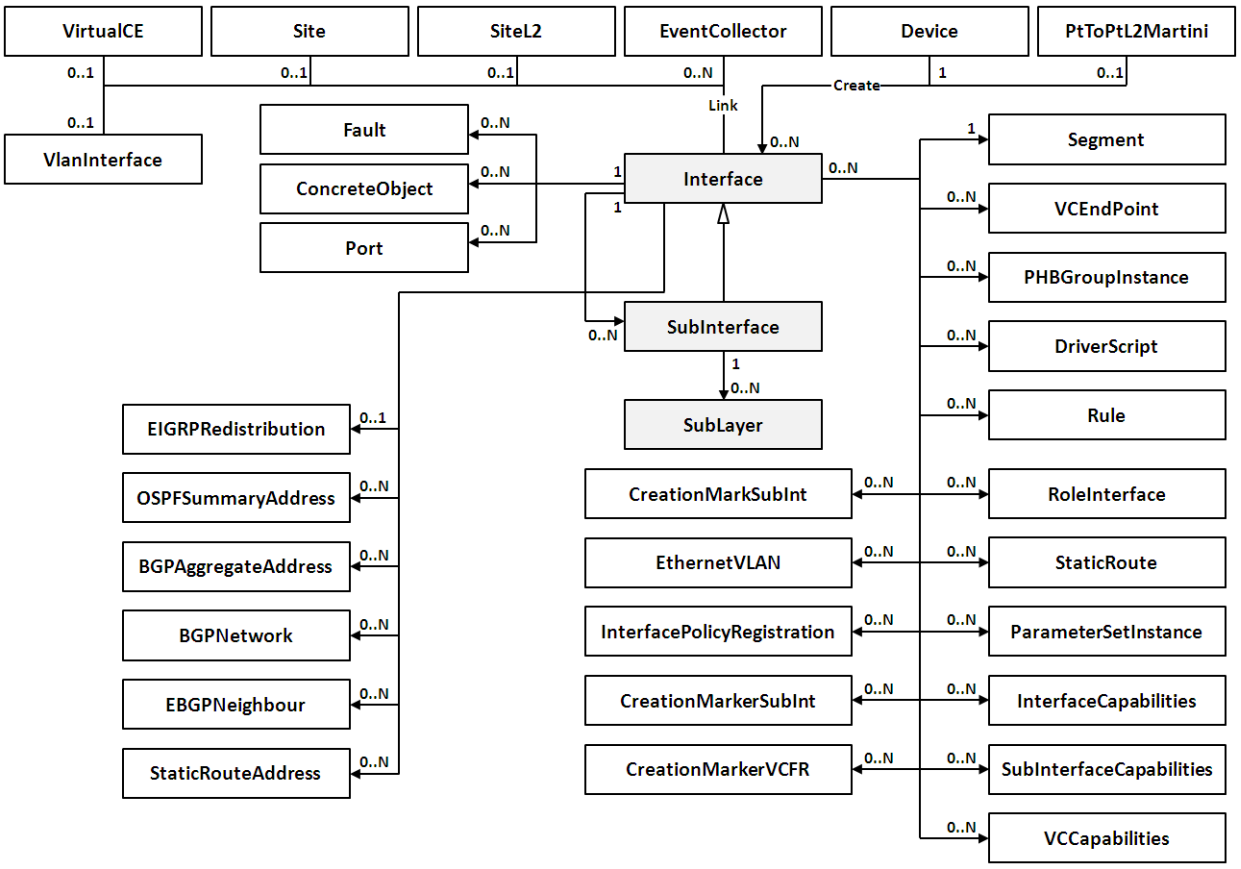


Table 4–92 describes the attributes for the Interface object.

Table 4–92 Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RCW	Name of the interface.
Description	String	[no default]	RCW	SNMP ifDescription.
Number	U32	0	RCW	Number of the interface (SNMP ifIndex).
Type	U32	1	RCW	SNMP ifType.
Hardware	String	[no default]	R	Hardware description
IpAddr	IPAddress	0.0.0.0	RCW	IP Address for interface if applicable.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
SubnetMask	IPAddress	0.0.0.0	RCW	Subnet mask.
Ipv6Addr	IPAddress	::/0	RCW	IPv6 Address and prefix length for the interface, if applicable.
PhysicalAddress	String	[no default]	RCW	Physical address of the interface (the MAC address).
Speed	U32	0	RCW	Speed of the interface in Kbits/s (SNMP ifSpeed parameter).
State	Enum	4	RCW	State of interface 0 = Down 1 = Up 2 = Testing 3 = Shutdown 4 = Unknown Constructed from the SNMP ifAdminStatus and ifOperStatus. Interfaces and sub-interfaces connected to a virtual device always have a state of "Unknown".
PublicPeIpAddr (Site only)	IPAddress	0.0.0.0	RO	PE interface address when not in a VRF.
PublicPeMask (Site Only)	IPAddress	0.0.0.0	RO	PE interface address when not in a VRF.
PrivatePeIpAddr (Site Only)	IPAddress	0.0.0.0	RO	PE interface address when in a VRF.
PrivatePeMask (Site only)	IPAddress	0.0.0.0	RO	PE interface address when in a VRF.
PrivateCeIpAddr (Site only)	IPAddress	0.0.0.0	RO	CE interface address when using EBGP and in a VRF.
PrivatePeIpv6Addr	IPAddress	::/0	RCW	PE interface Ipv6 address and prefix length when in a VRF.
PublicPeIpv6Addr	IPAddress	::/0	RCW	PE interface Ipv6 address and prefix length when not in a VRF.
PrivateInterface Description	String	[no default]	RO	Description of the interface.
VrfTableName (Site only)	String	[no default]	RO	The name of the VRF routing table.
OverrideVrfTable Limit (VPN site only)	Boolean	False	RO	True = Use site-specific settings for VRF table limits False = Use domain defaults for VRF table limits.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
VrfTableLimit (VPN site only)	U32	0	RO	Maximum number of routes allowed in a VRF (0=No limit).
VrfTableLimit Warning (VPN site only)	U32	0	RO	Percentage at which to warn of VRF table limits being exceeded. Range: 1–101, where 1–100 = percentage of VrfTableLimit reached warning. 101 = warning when VrfTableLimit reached
RDHighOrder (VPN site only)	U32	0	RO	The top 32 bits of the Route Descriptor value.
RDLowOrder (VPN site only)	U32	0	RO	The bottom 32 bits of the Route Descriptor value.
ActualRDHighOrder	U32	0	RO	The reference attribute that displays the actual RD value that EOM configures after VRF reduction and inheritance is taken into account. If you manually configure any RD values, you can use ActualRDHighOrder with ActualRDLowOrder to avoid duplication by keeping track of existing system RDs.
ActualRDLowOrder	U32	0	RO	The reference attributes that displays the actual RD value that EOM configures after VRF reduction and inheritance is taken into account. If you manually configure any RD values, you can use ActualRDLowOrder with ActualRDHighOrder to avoid duplication by keeping track of existing system RDs.
LocalPreference	U32	0xffffffff	RO	0xffffffff as default value in Cisco device this will equate to (100) when Local-preference command is not used. Can be in range 0 to *2^{32} - 2) for EBGP only. A lower value indicates a high priority.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
VrfExportFilter (Site only)	String	[no default]	RO	Name of the pre-existing VRF export filter. [no default] = no filter applied.
EbgpPrefixFilterIn (Site only)	String	[no default]	RO	Name of the pre-existing inbound prefix filter. [no default] = no filter applied.
EbgpPrefixFilterOut (Site only)	String	[no default]	RO	Name of the pre-existing outbound prefix filter. [no default] = no filter applied.
EbgpPrefixLimit (Site only)	U32	0	RO	The maximum number of eBGP prefixes that a PE is allowed to receive. Range: $1–2^{32}$ 0 = no limit
EbgpPrefixLimitWarning (Site only)	U32	0	RO	The percentage at which to warn of eBGP prefix limits being exceeded. Range: 1–101, where 1–100 = warning when percentage of EbgpPrefix limit reached. 101 = warning when EbgpPrefixLimit reached.
EbgpPrefixRestartDelay (Site only)	U32	0	RO	Delay, in minutes, before automatic restart of the EBGP session, after prefix limit was reached and session terminated. Range: 0–65535 0 = disabled
KeepAlive	U32	60	RO	Frequency, in seconds with which Keep Alive messages are sent to neighbors or a specific group of neighbors. Range: 0–65535 0 = disabled
HoldTime	U32	180	RO	Delay, in seconds. If a device does not receive a Keep Alive message until this time, it declares its neighbor to be dead. Range: 0–65535 0 = disabled
EnableAdvertisementInterval	Boolean	False	RO	Allows the exchange of eBGP routing updates between two neighbors.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
AdvertisementInterval	U32	0	RO	Delay, in seconds, after which the subsequent eBGP routing updates to be exchanged between two neighbors. Range: 0–600
UpdateSourceInterface	String	PE Interface ID	RCW	Allows the definition of source interface to be used for eBGP updates on per neighbor basis.
NeighbourDescription	String	Site Name	RCW	User defined description of neighbors on a per neighbor basis. Maximum 80 characters allowed.
AsOverride (Site only)	Boolean	False	RO	True = Set AS Override for EBGP neighbors. False = Do not set AS override.
OverrideAllowAsIn (Site only)	Boolean	False	RO	True = Site setting overrides the domain default setting of Allow AS in. False = Site setting should not override the domain default setting of Allow AS in.
AllowAsIn (Site only)	U32	0	RO	The number of times the same AS can appear in the AS path list. Range: 0–10
BgpMd5Key (Site only)	Encrypted string	[no default]	RO	BGP MD5 authentication key [no default] = not used.
EbgpMd5InheritFromVPN	Boolean	False	RO	True = inherit the interface's BGP authentication settings from the parent VPN. False = Does not inherit.
OverridePeCeSend Community (Site only)	Boolean	False	RO	True = use local settings for Send Community parameters. False = use domain-level settings.
PeCeSendStandard Community (Site only)	Boolean	False	RO	True = Use the PE-CE peering Standard send community tag. False = Does not use the PE-CE peering Standard send community tag.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
PeCeSendExtended Community (Site only)	Boolean	False	RO	True = Use the PE-CE peering Extended send community tag. False = Does not use the PE-CE peering Extended send community tag.
Context (Site only)	String	[no default]	RCW	Local context for driver scripts applied to interfaces (max 512 bytes).
VrfDesc (VPN site only)	String	[no default]	RO	Optional string describes the VRF applied to router, if the router supports VRF description string.
EIBgpMaxPaths (VPN site only)	U32	1	RO	Maximum number of multipaths on both EBGP and IBGP. Range: 1–6
IBgpUnequalCost	Boolean	False	RO	Allows load balancing by selecting iBGP paths that do not have equal cost.
VrfImport (VPN site only)	U32	1 (turned off)	RO	The number of device redundant path configurations. Range is 2^{16} plus default value.
UseVrfImport	Boolean	True	RO	True = Use VRF import False = Do not use VRF import
InternalName	String	[no default]	RCW	The internal name.
IsConfigurable	Boolean	[no default]	RO	Whether the interface is configurable.
IsController	Boolean	False	RC	Whether this interface has been tagged as controller during discovery
AutoInheritVrf	Boolean	False	RO	Try to inherit the VRF table parameters from the VRF template in the VPN this interface is connected to.
EBgpMaxPaths	U32	0	RO	Specification of the maximum number of parallel EBGP routes that can be installed in the routing table. This corresponds to the Cisco maximum-paths command. Range: 1–16
EbgpLocalAsn	U32	0	RO	Autonomous Systems path. Range: 1–4294967295
EbgpLocalAsnEnable	Boolean	False	RO	Whether Local Autonomous Systems Numbers is enabled or not.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
EbgpLocalAsnNoPrepend	Boolean	False	RO	If true, device does not prepend the local ASN to any routes received from the eBGP neighbor
EbgpLocalAsnReplaceAs	Boolean	False	RW	The replace-as keyword is used to prepend only the local autonomous-systemnumber (as configured with the ip-address argument) to the AS_PATH attribute.The autonomous-system number from the local BGP routing process is not prepended.
EbgpLocalAsnDualAs	Boolean	False	RW	The dual-as keyword is used to configure the eBGP neighbor to establish a peering session using the real autonomous-system number (from the local BGP routing process) or by using the autonomous-system number configured with the ip-address argument (local-as).
EbgpMd5InheritFromVPN	Boolean	False	RO	True = inherit the interface EBGP authentication settings from the parent VPN. False = Does not inherit the interface EBGP authentication settings.
EbgpMultihop	Boolean	False	RO	Enables BGP connections to devices on networks that are not directly connected (that are more than one hop away from a local device).
EbgpSoftReconfig	Boolean	False	RCW	Enables the EBGP soft reconfiguration setup command on Cisco and Juniper E-series devices. Setting this value does not issue a soft reconfiguration reset action - it enables the support for the reset action
IsNextHopSelf	Boolean	False	RCW	Enables configuration of the router as the next hop for a Border Gateway Protocol (BGP)-speaking neighbor. False = Don't generate the corresponding command.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
NeighborConnectionWeight	U32	0xffffffff	RCW	To assign a weight to a neighbor connection. Valid range is 0 – 65535. The value 0xffffffff(4294967295) used to specify no-value. This is equivalent to turning off the neighbor connection weight feature.
FilterListIn	U32	0xffffffff	RCW	Number of an autonomous system path access list for incoming routes. Valid range is 1 – 500. The value 0xffffffff (4294967295) used to specify no-value. This is equivalent to turning off the inbound filter-list feature.
FilterListOut	U32	0xffffffff	RCW	Number of an autonomous system path access list for incoming routes. Valid range is 1 – 500. The value 0xffffffff (4294967295) used to specify no-value. This is equivalent to turning off the outbound filter-list feature.
EbgpNeighborSoo	Boolean	False	RCW	True indicates that configuration of neighbor soo is required.
UseDefaultOriginateRouteMap	Boolean	False	RCW	True indicates that configuration of default-originate route map is required.
DefaultOriginateRouteMapName	Boolean	String	RCW	User can set this value to route-map name. This value will take into consideration only when UseDefaultOriginateRouteMap is true.
EigrpDampHalfLife	U32	False	RO	Time, in minutes, when a penalty applying to a route is decreased by half. Range: 1–45, default is 15.
EigrpDampMaxSuppressTime	U32	0	RO	Maximum time, in minutes, that a route can be suppressed. The valid range is 1–255 for Cisco and Juniper E-series devices, 1–720 for Juniper M-series devices. The default is 60 minutes.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
EigrpDampReuse	U32	0	RO	Reuse threshold (default 750) Range: 1–20 000
EigrpDampSuppress	U32	0	RO	Cutoff (suppression) threshold (default 2000) Range: 1–20 000
EigrpMaxPaths	U32	0	RO	Specification of the maximum number of parallel EIGRP routes that can be installed in the routing table. This corresponds to the Cisco maximum-paths command. Range: 1–16
EigrpMd5Enable	Boolean	0	RO	Enable MD5 key authentication for EIGRP for the interface
EigrpMd5InheritFromVPN	Boolean	0	RO	True = inherit the interface's EIGRP authentication settings from the parent VPN. False = Does not inherit.
EigrpMd5KeyChainRef	String	[no default]	RO	Key chain name to use with MD5 Authentication for EIGRP.
EnableAdvertisementInterval	Boolean	False	RO	EBGP: Allows the exchange of eBGP routing updates between two neighbors.
EnableExternalInboundRouteMap	Boolean	False	RO	Enables the use of a specified route map name for inbound external route-map. Disabled for virtual-CE.
UseExternalOutboundRouteMap	Boolean	False	RO	Enables specification of an outbound external BGP route-map for the interface
ExternalInboundRouteMap	String	[no default]	RO	Name of the inbound external route-map
ExternalOutboundRouteMap	String	[no default]	RO	Name of the outbound external route-map
InboundRouteMap	String	[no default]	RO	Inbound route map name.
InheritVRFRouteMapsFromVPN	Boolean	False	RO	Inherit the VRF table name and RD defined for the parent VPN
LoopbackIpAddr	IPAddress	0.0.0.0	RW	IP address for loopback.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
MultiVpnOverride	Boolean	False	RO	If a site is set to inherit VPN-wide VRF/RD details and participates in more than one VPN for which VPN-wide details are defined, IP Service Activator handles the conflict by applying site-specific automatically generated VRF/RD details to the site. If MultiVpnOverride and InheritVRFRouteMapsFrom VPN are both set to true: - if the site is a member of only one VPN, the VRF table name and RD are derived from the parent VPN - if the site is a member of multiple VPNs, the VRF table name and RD are derived using the site specific options
NoRedistributeStaticMerge	Boolean	False	RW	Use Default Static Route Redistribution
NoRedistributeBgp2Eigrp	Boolean	FALSE	RO	Used to turn off route redistribution from Bgp to Eigrp; when this attribute is set to true, routes will not be redistributed from Bgp to Eigrp. For this attribute, access rights change after this interface is linked to the site and connectivity is set as BGP.
NoRedistributeBgp2Ospf	Boolean	FALSE	RO	Used to turn off route redistribution from Bgp to Ospf; when this attribute is set to true, routes will not be redistributed from Bgp to Ospf. By default access will be "RO", when the interface is linked to some site and if the connectiviy changed to BGP then this attribute access changes to "RCW".

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
NoRedistributeEigrp2Bgp	Boolean	FALSE	RO	Used to turn off route redistribution from Eigrp to Bgp; when this attribute is set to true, routes will not be redistributed from Eigrp to Bgp. Access is R only by default. For this attribute access rights will change once this interface is linked to the site and connectivity is EIGRP.
NoRedistributeEigrp2Ospf	Boolean	FALSE	RO	Used to turn off route redistribution from Eigrp to Ospf; when this attribute is set to true, routes will not be redistributed from Eigrp to Ospf. Access is RO by default. For this attribute access rights will change once this interface is linked to the site and connectivity is EIGRP.
NoRedistributeEigrp2Rip	Boolean	FALSE	RO	Used to turn off route redistribution from Eigrp to Rip; when this attribute is set to true, routes will not be redistributed from Eigrp to Rip. Access for attribute NoRedistributeEigrp2Rip is RO only by default. For this attribute access rights will change once this interface is linked to the site and connectivity is EIGRP.
NoRedistributeOspf2Bgp	Boolean	FALSE	RO	Used to turn off route redistribution from Ospf to Bgp; when this attribute is set to true, routes will not be redistributed from Ospf to Bgp. Access type is not 'RW' by default for NoRedistributeOspf2Bgp attribute (i.e. Access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is OSPF.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
NoRedistributeOspf2Eigrp	Boolean	FALSE	RO	Used to turn off route redistribution from Ospf to Eigrp; when this attribute is set to true, routes will not be redistributed from Ospf to Eigrp. Access type is not 'RW' by default for NoRedistributeOspf2Eigrp attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is OSPF.
NoRedistributeOspf2Rip	Boolean	FALSE	RO	Used to turn off route redistribution from Ospf to Rip; when this attribute is set to true, routes will not be redistributed from Ospf to Rip. Access type is not 'RW' by default for NoRedistributeOspf2Rip attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is OSPF.
NoRedistributeRip2Eigrp	Boolean	FALSE	RO	Used to turn off route redistribution from Rip to Eigrp; when this attribute is set to true, routes will not be redistributed from Rip to Eigrp. Access type is not 'RW' by default for NoRedistributeRip2Eigrp attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is RIP.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
NoRedistributeRip2Ospf	Boolean	FALSE	RO	Used to turn off route redistribution from Rip to Ospf; when this attribute is set to true, routes will not be redistributed from Rip to Ospf. Access type is not 'RW' by default for NoRedistributeRip2Ospf attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is RIP.
NoRedistributeStatic2Bgp	Boolean	FALSE	RO	Used to turn off route redistribution from Static to Bgp; when this attribute is set to true, routes will not be redistributed from Static to Bgp. Access type is not 'RW' by default for NoRedistributeStatic2Bgp attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is RIP/OSPF/EIGRP.
NoRedistributeStatic2Eigrp	Boolean	FALSE	RO	Used to turn off route redistribution from Static to Eigrp; when this attribute is set to true, routes will not be redistributed from Static to Eigrp. Access type is not 'RW' by default for NoRedistributeStatic2Eigrp attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is RIP/OSPF/EIGRP.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
NoRedistributeStatic2Ospf	Boolean	FALSE	RO	Used to turn off route redistribution from Static to Ospf; when this attribute is set to true, routes will not be redistributed from Static to Ospf. Access type is not 'RW' by default for NoRedistributeStatic2Ospf attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is RIP/OSPF/EIGRP.
NoRedistributeStatic2Rip	Boolean	FALSE	RO	Used to turn off route redistribution from Static to Rip; when this attribute is set to true, routes will not be redistributed from Static to Rip. Access type is not 'RW' by default for NoRedistributeStatic2Rip attribute (i.e. access is R only by default). For this attribute access rights will change once this interface is linked to the site and connectivity is RIP/OSPF/EIGRP.
OspfArea	IPAddress OrInt	0	RO	OSPF area ID for the selected interface
OspfAreaIsIpAddress	Boolean	False	RO	Whether OspfArea is specified as an IP Address or as an integer
OspfCost	U32	0	RO	Cost of sending a packet on the selected interface. Range: 1–65535
OspfAccessListName	String	[no default]	RW	Named ACL for Distribute In filtering. The access list specified is applied to incoming updates on the selected interface.
OspfDistributeOutAcl	String	[no default]	RO	Named ACL for Distribute Out filtering. The access list specified is applied to outgoing updates on the selected interface and suppresses networks from being advertised in updates.
OspfMaxPaths	U32	0	RO	Maximum redundant routes OSPF can use. Range: 1–6

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
OspfMd5AuthLocally	Boolean	False	RO	Enable MD5 key authentication for OSPF locally on the interface.
OspfMd5InheritFromVPN	Boolean	False	RO	Inherit the interface's OSPF authentication settings from the parent VPN.
OspfNSSANoRedistribution	Boolean	False	RO	Suppression of the NSSA behavior in which Type 7 LSAs are translated to Type 5 LSAs.
OspfProcessID	U32	0	RO	OSPF process ID for the selected interface. Range:1-65535
OspfRedistTagValueFromBgp	U32	0	RO	Tag value to identify routes redistributed into OSPF from BGP.
OspfSpfHold	U32	0	RO	Minimum time in milliseconds between consecutive SPF recalculations.
OspfSpfMaxWait	U32	0	RO	Maximum wait time in milliseconds between consecutive SPF recalculations.
OspfSpfStart	U32	[no default]	RO	Minimum delay in milliseconds between the reception of a topology change and the start of SPF recalculation.
OspfUseRedistTagFromBgp	Boolean	False	RO	Enable the use of the tag value to identify routes redistributed into OSPF from BGP.
OspfUseSpfThrottling	Boolean	False	RO	Enable control of timing and execution of SPF recalculations.
OverridePeCeSendCommunity	Boolean	False	RO	Override PE-CE community bit sharing
OverrideVrfTableLimit	Boolean	False	RO	Override the domain VRF table limit
PeCeSendExtendedCommunity	Boolean	False	RO	Share extended community between PE and CE devices on a VPN
PeCeSendStandardCommunity	Boolean	False	RO	Standard community between PE and CE devices on a VPN
PolicyServiceCustomer	String	[no default]	RO	Name of the policy service customer.
PrivateInterfaceDescription	String	[no default]	RO	Description of the interface; such as the name of the customer that is associated with the interface.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
PrivatePeIpAddrI[no default]	Boolean	False	RO	Whether private PE IP address is allocated by I[no default].
PrivatePeIpUnnumberedRef	String	[no default]	RO	Interface name to use for IP unnumbered Private PE addressing. This allows to enable IP on an interface and use it in a VPN without having to assign an explicit Private PE IP address and mask. Instead, the IP address of loopback address from the device is used.
PrivatePeIpv6UnnumberedRef	String	[no default]	RCW	Interface name to use for IPv6 unnumbered Private PE addressing. This allows to enable IPv6 on an interface and use it in a VPN without having to assign an explicitPrivate PE IPv6 address. Instead, the IPv6 address of loopback address from the device is used.
UseExternalInboundRouteMap (Site only)	Boolean	False	RO	True = Use the specified external inbound route-map. False = Do not use the specified external inbound route map. **Note**: Use a naming scheme different from IP Service Activator's for external inbound and outbound route-maps. IP Service Activator will remove route-maps with the same naming as those which it generates when the device is unmanaged and re-managed.
ExternalInboundRouteMap (Site only)	String	[no default]	RO	Value for the external inbound route-map.
UseExternalOutboundRouteMap (Site only)	Boolean	False	RO	True = Use the specified external outbound route-map. False = Do not use the specified external outbound route map.
ExternalOutboundRouteMap (Site only)	String	[no default]	RO	Value for the external outbound route-map.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
EBgpDampHalfLife (VPN site only)	U32	0	RO	Time, in minutes, at which a penalty applying to a route is decreased by half. Range: 1–45 0 = No EBGP dampening. 15 = Default if EBGP dampening applied. **Note**: IP Service Activator does not support EBGP Dampening on Network Processor cartridge managed devices. Refer to the respective Cartridge guide for details on the device capabilities supported by the cartridge.
EBgpDampMaxSuppressTime (VPN site only)	U32	60	RO	Max. time, in minutes, that a route can be suppressed. in minutes. Range: 1–720
EBgpDampSuppress (VPN site only)	U32	2000	RO	A route is suppressed when its penalty exceeds this limit. Range: 1 –20 000. Must be greater than or equal to EBgpDampReuse.
EBgpDampReuse (VPN site only)	U32	750	RO	When the penalty applying to a route falls below this value, the route is unsuppressed. Range: 1–20 000
RedistributeDefault Route (Site only)	Boolean	False	RW	True = Redistribute the Default route. False = Does not redistribute the Default route.
RedistributeConnected (Site only)	Boolean	False	RO	True = Redistribute the connected routes. False = Does not redistribute the connected routes.
UseDefaultRedistribution (Site only)	Boolean	False	RO	True = use default redistribution metrics and policies and import RIP metric from VPN. False = Does not use default.
ForceVrfInstall (Site only)	Boolean	False	RO	True = VRF tables on corresponding interfaces must be installed and cannot be merged into other tables. False= VRF tables can be merged into other tables.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
ShareableVrf (Site only)	Boolean	False	RO	True = Other tables can be merged into this VRF table. False = Other tables cannot be merged into this VRF table.
VrfExportFilter (Site only)	String	[no default]	RO	Value for the external VRF import map name
VrfImportFilter (Site only)	String	[no default]	RO	Value for the external VRF import map name [no default] = not applicable; no default is used
RedistMetricConnected2Bgp	U32	0	RO	Values for redistribution metric:distribution into BGP. Range is 0 to 2^{32}-1 (VPN site only)
RedistMetricStatic2Bgp	U32	0	RO	Values for redistribution metric:distribution into BGP. Range is 0 to 2^{32}-1 (VPN site only)
RedistMetricRip2Bgp	U32	0	RO	Values for redistribution metric:distribution into BGP. Range is 0 to 2^{32}-1 (VPN site only)
RedistMetricOspf2Bgp	U32	0	RO	Values for redistribution metric:distribution into BGP. Range is 0 to 2^{32}-1 (VPN site only)
RedistMetricEigrp2Bgp	U32	0	RO	Values for redistribution metric:distribution into BGP. Range is 0 to 2^{32}-1 (VPN site only)
RedistPolicyConnected2Bgp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into BGP. (VPN site only)
RedistPolicyStatic2Bgp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into BGP. (VPN site only)
RedistPolicyRip2Bgp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into BGP. (VPN site only)

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistPolicyOspf2Bgp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into BGP. (VPN site only)
RedistPolicyEigrp2Bgp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into BGP. (VPN site only)
RedistMetric Connected2Rip	U32	0	RO	Values for redistribution metric: distribution into RIP. Range: 0–16 (VPN site only)
RedistMetricStatic2Rip	U32	0	RO	Values for redistribution metric: distribution into RIP. Range: 0–16 (VPN site only)
RedistMetricOspf2Rip	U32	2	RO	Values for redistribution metric: distribution into RIP. Range: 0–16 (VPN site only)
RedistMetricBgp2Rip	U32	1	RO	Values for redistribution metric: distribution into RIP. Range: 0–16 (VPN site only)
RedistMetricEigrp2Rip	U32	0	RO	Values for redistribution metric: distribution into RIP. Range: 0–16 (VPN site only)
RedistPolicyEigrp2Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into RIP. (VPN site only)
RedistPolicyConnected2 Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into RIP. (VPN site only)
RedistPolicyStatic2Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into RIP. (VPN site only)
RedistPolicyOspf2Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into RIP. (VPN site only)

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistPolicyBgp2Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into RIP. (VPN site only)
RedistMetricConnected2Ospf	U32	1	RO	Values for redistribution metric: distribution into OSPF. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricStatic2Ospf	U32	0	RO	Values for redistribution metric: distribution into OSPF. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricRip2Ospf	U32	2	RO	Values for redistribution metric: distribution into OSPF. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricBgp2Ospf	U32	1	RO	Values for redistribution metric: distribution into OSPF. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricEigrp2Ospf	U32	20	RO	Values for redistribution metric: distribution into OSPF. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricTypeConnected2Ospf	String	"2"	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeStatic2Ospf	String	"2"	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeRip2Ospf	String	"2"	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeBgp2Ospf	String	"2"	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeDefault2Ospf	String	"2"	RO	Value "1" or "2". (VPN site only)
RedistPolicyConnected2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistPolicyStatic2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistPolicyRip2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistPolicyBgp2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistPolicyEigrp2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistMetricRip2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricBgp2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricConnected2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricOspf2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricStatic2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricConnected2Bgp	U32	0	RO	Values for redistribution metric:distribution into BGP. Range is 0 to 2^{32}-1 (VPN site only)

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistMetricConnected2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricStatic2Eigrp	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricConnected2Ospf	U32	0	RO	Values for redistribution metric: distribution into OSPF. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricDefault2Ospf	U32	0	RO	Values for redistribution metric: distribution into EIGRP. Range is 0 to 2^{24}-2 (VPN site only)
RedistMetricTypeBgp2Ospf	String	[no default]	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeConnected2Ospf	String	[no default]	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeDefault2Ospf	String	[no default]	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeRip2Ospf	String	[no default]	RO	Value "1" or "2". (VPN site only)
RedistMetricTypeStatic2Ospf	String	[no default]	RO	Value "1" or "2". (VPN site only)
RedistPolicyConnected2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistPolicyDefault2Ospf	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistPolicyConnected2Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistPolicyDefault2Rip	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into OSPF. (VPN site only)
RedistributeBgp2Rip	Boolean	False	RO	Redistribute BGP Routes into RIP
RedistributeRip2Bgp	Boolean	False	RO	Redistribute RIP Routes into BGP
RedistributeRip2Bgp	Boolean	False	RO	Redistribute Connected routes (Ospf / Rip)
RedistributeDefaultRoute	Boolean	False	RO	Redistribute Default route
RedistributeDefaultRoute Ospf	Boolean	False	RO	Redistribute Default routes (OSPF)
RedistributeDefaultRouteRip	Boolean	False	RO	Redistribute Default routes (RIP)
RedistPolicyRip2Eigrp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into EIGRP. (VPN site only)
RedistPolicyBgp2Eigrp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into EIGRP. (VPN site only)
RedistPolicyConnected2Eigrp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into EIGRP. (VPN site only)
RedistPolicyOspf2Eigrp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into EIGRP. (VPN site only)
RedistPolicyStatic2Eigrp	String	[no default]	RO	Values for redistribution routemap name (policy): distribution into EIGRP. (VPN site only)
RemovePrivateAs	Boolean	False	RO	EBGP: turns on the removal of private autonomous system (AS) numbers from the autonomous system paths advertised by the neighbor WAN Address
UseDefaultRedistribution	Boolean	False	RO	When True, the default value will be used in Metrics and Policy fields for redistribution of protocols (EBGP, OSPF, RIP, and EIGRP).

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
UseVrfLabel	Boolean	False	RO	Enable vrf-table-label support on Juniper devices. When this value is set to true, the inner (VPN) label of a packet is removed as it arrives at a VRF so that it can be processed based on the contents of its IP header. When set to false, incoming packets are mapped directly onto an outgoing (CE-facing) interface based on the inner VPN label.
ttl	U32	[no default]	RO	0 = ignored Range: 1–255 Only applicable if EbgpMultihop is true.
InstallDhcp	Boolean	False	RW	True = Install DHCP support on the VRFs. (When applied at Interface level, overrides per-VPN settings.) False = Do not install DHCP support on VRFs.
PrimaryDhcpIpAddr	IPAddress	0.0.0.0	RW	Primary DHCP Server
SecondaryDhcpIpAddr	IPAddress	0.0.0.0	RW	Secondary DHCP Server
RIPIgnoreRoutes (VPN site only)	Boolean	False	RO	True=RIP routes from the specified IP address and mask are to be ignored. False=Do not ignore routes.
RIPIgnoreRoutesAddress (VPN site only)	IPAddress	0.0.0.0	RO	IP Address from which to ignore routes
RIPIgnoreRoutesMask (VPN site only)	IPAddress	0	RO	Mask of IP Address from which to ignore routes
RIPPassiveInterface (VPN site only)	Boolean	False	RO	True=Interface is configured as a passive (i.e. listen only) interface False=Not a passive interface
OSPFAreaType (VPN site only)	Enum	0	RO	0 = Normal 1 = Nssa 2 = NssaTotallyStub 3 = Stub 4 = StubTotallyStub
OspfMd5Key (VPN site only)	Encrypted string	[no default]	RO	OSPF MD5 authentication key [no default]=not used.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
OspfMd5InheritFromVPN (VPN site only)	Boolean	False	RO	True = inherit the interface's OSPF authentication settings from the parent VPN. False = Don't.
OspfNSSANoRedistribution (VPN site only)	Boolean	False	RO	True = suppresses the NSSA behavior in which Type 7 LSAs are translated to Type 5 LSAs. False = Does not suppress the translation behavior.
BgpAsn (Virtual CE only)	U32	0	RO	BGP Autonomous System Number. Unique number for routing.
RoutingProtocol (Virtual CE only)	Enum	3	RO	Type of routing being used between the PE and CE, relevant to MPLS VPNs only: 0 = EBGP 1 = RIP 3 = None 4 = OSPF 5 = EBGP_OSPF 6 = EBGP_RIP 7 = EIGRP 8 = EBGP_EIGRP
InstallStatic (Virtual CE only)	Boolean	True	RO	True = Static routing is used in conjunction with relevant routing protocol. False = Static routing is not used.
InstallLocalStatic This attribute is now unused.	Boolean	True	RO	True = Static routes defined in the site are not redistributed. False = Static routes defined in the site are redistributed.
InheritRouting	Boolean	False	RO	Always True for interfaces in VPN sites, so that the routing protocol and related attributes are inherited from the site object. Always False for VirtualCEs, so the routing protocol can be specified per interface.

Table 4–92 (Cont.) Interface Object Attributes

Attribute Name	Type	Default	Access	Explanation
EffectiveCommandDeliveryMode (This value is inherited when the command delivery mode is set at the Device or parent Interface level.)	Enum	0	RO	0 = Online: commands delivered to device 1 = OfflineMaintenance: commands not delivered, concretes and other states updated 2 = OfflineTest: commands not delivered
CommandDeliveryMode (This value is set when the command delivery mode is set at the Interface or SubInterface level.)	Enum	0	RCW	0 = Online: commands delivered to device 1 = OfflineMaintenance: commands not delivered, concretes and other states updated
IsConnectionModeEnabled	Boolean	False	RCW	Enables the type of connection
ConnectionModeType	Enum	0	RCW	0=Active 1=Passive
IsPathMTUDiscoveryEnabled	Boolean	False	RCW	Enables TCP transport path maximum transmission unit (MTU) discovery
IsSessionOptionEnabled	Boolean	False	RCW	Enables TCP transport session for address families
SessionOptionType	Enum	0	RCW	0=Single 1=Multi

Port Object

The Port is used to represent an Interface IP address and CIDR mask (IPv4) or prefix length (IPv6). Port objects are typically created during device discovery. An Interface may have multiple associated Port objects, but only one Port object in each address family will be considered the primary, or default, Port for that interface. The primary IPv4 Port address, if present, appears in the Interface object as the IpAddr and SubnetMask attributes. The primary IPv6 Port address, if present, appears in the Interface object as the Ipv6Addr attribute.

Table 4–93 describes the attributes for the Port object.

Table 4–93 Port Object Attributes

Attribute Name	Type	Default	Access	Explanation
IpAddr	IPAddress	"0.0.0.0/0"	RCW	Port IP address. Includes the CIDR mask (IPv4) or prefix length (IPv6).
Primary	Boolean	False	RCW	Indicates whether the port represents the primary, or default, IP address for the associated interface in the specified address family. An interface may have only one primary port in each address family.

Table 4–93 (Cont.) Port Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	IP address without the CIDR mask (IPv4) or prefix length (IPv6).

EigrpRedistribution Object

Redistribution attributes (delay, reliability, loading and mtu) from other protocols (connected, static, Bgp, Rip) into Eigrp.

Table 4–94 describes the attributes for the EigrpRedistribution object.

Table 4–94 EigrpRedistribution Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistDelayFromBgp	U32	4294967295	RCW	Specify Delay in tens of microseconds. Default = 1000 (This is 10 milliseconds)
RedistDelayFromConnected	U32	4294967295	RCW	Specify Delay in tens of microseconds. Default = 1000 (This is 10 milliseconds)
RedistDelayFromRip	U32	4294967295	RCW	Specify Delay in tens of microseconds. Default = 1000 (This is 10 milliseconds)
RedistDelayFromStatic	U32	4294967295	RCW	Specify Delay in tens of microseconds. Default = 1000 (This is 10 milliseconds)
RedistLoadingFromBgp	U32	4294967295	RCW	Specify the effective load on the link. Range: 0–255, where 255 is 100% loading. Default = 1
RedistLoadingFromConnected	U32	4294967295	RCW	Specify the effective load on the link. Range: 0–255, where 255 is 100% loading. Default = 1
RedistLoadingFromRip	U32	4294967295	RCW	Specify the effective load on the link. Range: 0–255, where 255 is 100% loading. Default = 1
RedistLoadingFromStatic	U32	4294967295	RCW	Specify the effective load on the link. Range: 0–255, where 255 is 100% loading. Default = 1
RedistMtuFromBgp	U32	4294967295	RCW	Maximum Transmission Unit of the path in bytes. Default = 1500 (typical for Ethernet interface.)

Table 4–94 (Cont.) EigrpRedistribution Object Attributes

Attribute Name	Type	Default	Access	Explanation
RedistMtuFromConnected	U32	4294967295	RCW	Maximum Transmission Unit of the path in bytes. Default = 1500 (typical for Ethernet interface.)
RedistMtuFromRip	U32	4294967295	RCW	Maximum Transmission Unit of the path in bytes. Default = 1500 (typical for Ethernet interface.)
RedistMtuFromStatic	U32	4294967295	RCW	Maximum Transmission Unit of the path in bytes. Default = 1500 (typical for Ethernet interface.)
RedistReliabilityFromBgp	U32	4294967295	RCW	Default = 255, represents 100% reliability.
RedistReliabilityFromConnected	U32	4294967295	RCW	Default = 255, represents 100% reliability.
RedistReliabilityFromRip	U32	4294967295	RCW	Default = 255, represents 100% reliability.
RedistReliabilityFromStatic	U32	4294967295	RCW	Default = 255, represents 100% reliability.
Name	String		RO	

OspfSummaryAddress Object

The OspfSummaryAddress is used to configure the advertising of OSPF routes for redistribution as a summary address. It is an aggregate list of addresses represented by a single IP address and subnet mask.

The use of summary addressing reduces the overhead incurred to manage the link-state database. Instead of advertising all OSPF routes encompassed by the summary address, a single summary route is advertised. The use of this configuration corresponds to the following command:

```
Router(config-router)# summary-address <ip-address mask> | <prefix-mask>
[not-advertise] [tag <tag>]
```

Each interface in the list can be configured with a summary address list. The SuppressAdvertise attribute configures the site not to advertise the OSPF routes encompassed by the summary address and subnet mask. In other words, these routes are filtered out. This corresponds to the use of the not-advertise flag in the Cisco IOS command above.

The tag parameter is supported by the UseTag attribute.

Table 4–95 describes the attributes of the OspfSummaryAddress object.

Table 4–95 OspfSummaryAddress Object Attributes

Attribute Name	Type	Default	Access	Explanation
IpAddr	IPAddress	[no default]	RCW	The base address to be used as the summary address.

Table 4–95 (Cont.) OspfSummaryAddress Object Attributes

Attribute Name	Type	Default	Access	Explanation
SubnetMask	IPAddress	[no default]	RCW	Use this field to specify the number of set prefix mask bits. 32 corresponds to 255.255.255.255
SuppressAdvertise	Boolean	False	RCW	To specify the 'no' form of the summary-address command. When this is set to true, the summary addresses specified will not be advertised.
TagValue	U32	[no default]	RCW	Tag value to be used as the tag parameter in the summary-address command configured on the device.
UseTag	Boolean	False	RCW	Enables the use of TagValue as the tag parameter in the summary-address command configured on the device.
Name	String	Content of IpAddr used as a string	RO	Content of IpAdd

BgpAggregateAddress

Table 4–96 describes the attributes for BgpAggregateAddress.

Table 4–96 BgpAggregateAddress Attributes

Attribute Name	Type	Default	Access	Explanation
IpAddr	IPAddress	[no default]	RCW	Aggregated IP address for the route summary that BGP will advertise.
SubnetMask	IPAddress	[no default]	RCW	Aggregated subnet mask for the route summary that BGP will advertise.
Name	String	Content of IpAddr used as a string	RO	BGP aggregate address name.

BgpNetwork

Table 4–97 describes the attributes for BgpNetwork.

Table 4–97 BgpNetwork Attributes

Attribute Name	Type	Default	Access	Explanation
IpAddr	IPAddress	[no default]	RCW	IP address for the network that BGP will advertise.
SubnetMask	IPAddress	[no default]	RCW	Subnet mask for the network that BGP will advertise.
Name	String	Content of IpAddr used as a string	RO	BGP network attribute name.

EbgpNeighbour Object

The EbgpNeighbour object represents a single CE peer device when defining a CE-PE relationship during VPN configuration. A PE Interface object may be associated with one or more EbgpNeighbour objects when the interface is linked to a VPN Site object and the site routing protocol is defined as one of EBGP, EBGP_OSPF, EBGP_RIP, or EBGP_EIGRP. The EbgpNeighbour address must be in the same address family as one of the Interface private PE addresses. In other words, an IPv4 EbgpNeighbour must have a corresponding IPv4 private PE IP address and an IPv6 EbgpNeighbour must have a corresponding private PE IPv6 address. The attributes of the first EbgpNeighbour appear as attributes of the parent Interface object.

Table 4–98 describes the attributes for the EbgpNeighbour object.

Table 4–98 EbgpNeighbour Object Attributes

Attribute Name	Type	Default	Access	Explanation
AdvertiseAddress Family	Enum	Unspecified	RCW	Specifies the BGP address family in which to advertise the neighbour. Possible values include Unspecified: advertise in the same address family as the neighbor IP address. IPv4: advertise in the IPv4 address family. IPv6: advertise in the IPv6 address family. Both: advertise in both the IPv4 and IPv6 address family
AdvertisementInterval	U32	0	RCW	Delay, in seconds, after which the subsequenet eBGP routing updates are exchanged between two neighbors. Range: 0-600
AllowAsIn	U32	0	RCW	The number of times the same AS can appear in the AS path list. Range: 0-10
AsOverride	Boolean	False	RCW	True = Set AS override for eBGP neighbors. False = Do not set AS override.
BgpAsn	U32	0	RCW	BGP Autonomous System Number. Unique number for routing.
BgpMd5Key	String	""	RCW	BGP MD5 authentication key

Table 4–98 (Cont.) EbgpNeighbour Object Attributes

Attribute Name	Type	Default	Access	Explanation
ConnectionModeType	Enum	Active	RCW	Specifies the type of connection for an enabled TCP transport session. Possible values include Active Passive
DefaultOriginateRouteMapName	String	""	RCW	Specifies the default originating route-map name. Only applies if UseDefaultOriginateRouteMapName is set to True.
EbgpLocalAsn	U32	0	RCW	Local Autonomous System Number. Range: 1-4294967295
EbgpLocalAsnEnable	Boolean	False	RCW	Determines whether local ASN is enabled.
EbgpLocalAsnNoPrepend	Boolean	False	RCW	If true, the device dos not prepend the local ASN to any routes received from the eBGP neighbor.
EbgpMd5InheritFromVPN	Boolean	False	RCW	If true, inherit the neighbor's eBGP authentication settings from the parent VPN.
EbgpMultihop	Boolean	False	RCW	Enables BGP connections to devices on networks that are not directly connect, i.e., that are more than one hop away from a local device.
EbgpNeighborSoo	Boolean	False	RCW	If true, configuration of the neighbor SOO is required.
EbgpPrefixFilterIn	String	""	RCW	Name of the pre-existing inbound prefix filter. If unset, no filter is applied.
EbgpPrefixFilterOut	String	""	RCW	Name of the pre-existing outbound prefix filter. If unset, no filter is applied.
EbgpPrefixLimit	U32	0	RCW	The maximum number of eBGP prefixes that a PE is allowed to receive. Range: 1-4294967295 0 = no limit.

Table 4–98 (Cont.) EbgpNeighbour Object Attributes

Attribute Name	Type	Default	Access	Explanation
EbgpPrefixLimit Warning	U32	0	RCW	The percentage at which to warn of eBGP prefix limits being exceeded. Range: 1-101, where 1-100 = warning when percentage of eBGP prefix limit reached. 101 = warning when eBGP prefix limit reached.
EbgpPrefixRestart Delay	U32	0	RCW	Delay, in minutes, before automatic restart of the eBGP session, after the prefix limit was reached and the session terminated. Range: 0-65535 0 = disabled
EbgpSoftReconfig	Boolean	False	RCW	Enables the eBGP soft reconfiguration setup command on Cisco and Juniper E-series devices. Setting this value does not issue a soft reconfiguration reset action - it enables the support for the reset action.
EnableAdvertise mentInterval	Boolean	False	RCW	Allows the exchange of eBGP routing updates between two neighbors.
EnableExternalIn boundRouteMap	Boolean	False	RCW	Enables the use of a specified route map name for inbound external route-map. Disabled for Virtual-CE.
ExternalInbound RouteMap	String	""	RCW	Name of the inbound external route-map.
ExternalOutboun dRouteMap	String	""	RCW	Name of the outbound external route-map.
FilterListIn	U32	0xffffffff	RCW	Number of an autonomous system path access-list for incoming routes. Range: 1-500 0xffffffff (4294967295) specifies no value. This is equivalent to disabling the incoming filter list feature.

Table 4–98 (Cont.) EbgpNeighbour Object Attributes

Attribute Name	Type	Default	Access	Explanation
FilterListOut	U32	0xffffffff	RCW	Number of an autonomous system path access-list for incoming routes. Range: 1-500 0xffffffff (4294967295) specifies no value. This is equivalent to disabling the incoming filter list feature.
HoldTime	U32	180	RCW	Interval, in seconds, to wait for a keep-alive message. If a keep-alive has not been received in this time period, the device will declare the neighbor dead. Range: 0-65535 0 = disabled
InboundRouteMap	String	""	RCW	Name of the inbound route-map.
IpAddr	IPAddress	0.0.0.0	RCW	eBGP neighbor IP address. This must be in the same address family as one of the PE interface private IP addresses.
IsConnectionModeEnabled	Boolean	False	RCW	Determines whether the ConnectionModeType attribute is used.
IsNextHopSelf	Boolean	False	RCW	Enables configuration of the router as the next hop for a BGP neighbor. False = do not generate the corresponding command.
IsPathMTUDiscoveryEnabled	Boolean	False	RCW	Enables TCP transport path MTU discovery.
IsSessionOptionEnabled	Boolean	False	RCW	Enables TCP transport session.
KeepAlive	U32	60	RCW	Frequency, in seconds, with which keep-alive messages are sent to neighbors or a specific group of neighbors. Range: 0-65535 0 = disabled

Table 4–98 (Cont.) EbgpNeighbour Object Attributes

Attribute Name	Type	Default	Access	Explanation
LocalPreference	U32	0xffffffff	RCW	When configuring a route-map, specify a preference value for the autonomous system path. Range: 0-0xffffffff (4294967295) 0xffffffff disables the command
NeighborConnectionWeight	U32	0xffffffff	RCW	Assign a weight to a neighor connection. Range: 0-65535 0xffffffff specifies no value, which is equivalent to turning off the neighbor connect weight feature.
NeighbourDescription	String	""	RCW	User-defined description of the neighbor. Maximum 80 characters allowed.
OverrideAllowAsIn	Boolean	False	RCW	If true, AS override is set for eBGP this eBGP neighbor.
OverridePeCeSendCommunity	Boolean	False	RCW	If true, use local settings for Send Community parameters. If false, use domain-level settings.
PeCeSendExtendedCommunity	Boolean	False	RCW	If true, use the PE-CE peering Extended send community tag.
PeCeSendStandardCommunity	Boolean	False	RCW	If true, use the PE-CE peering Standard send community tag.
RemovePrivateAs	Boolean	False	RCW	If true, the removal of private ASNs from the autonomous system paths advertised by the neighbor WAN address is enabled.
SessionOptionType	Enum	Single	RCW	Determines the type of TCP transport session. Possible values include Single = all address families use a single TCP session. Multi = separate TCP session for each address family.
ttl	U32	0	RCW	eBGP multi-hop time-to-live in hops. Range: 0-255 0 = ignored

Table 4–98 (Cont.) EbgpNeighbour Object Attributes

Attribute Name	Type	Default	Access	Explanation
UpdateSourceInterface	String	""	RO	If UseUpdateSourceIfDefault is true, this will display the name of the associated PE interface. The associated BGP sessions will use this interface for TCP connections.
UpdateSourceInterfaceRef	String	""	RCW	If set, this will override the value in UpdateSourceInterface. This allows the associated BGP session to use an arbitrary PE interface for TCP connections. If not set and UseUpdateSourceIfDefault is false, the BGP session will use the closest or best local interface.
UseDefaultOriginateRouteMap	Boolean	False	RCW	If true, use the DefaultOriginateRouteMap name.
UseExternalInboundRouteMap	Boolean	False	RCW	If true, use the external inbound route-map specified in ExternalInboundRouteMap.
UseExternalOutboundRouteMap	Boolean	False	RCW	If true, use the external outbound route-map specified in ExternalOutboundRouteMap.
UseNeighbourDescDefault	Boolean	False	RCW	If true and no user-defined value is available for NeighbourDescription, the name of the associated VPN site will be used as the eBGP neighbor description.
UseUpdateSourceIfDefault	Boolean	False	RCW	If true, the associated PE interface will be used for the associated BGP session's TCP connections. The PE interface will be set in UpdateSourceInterface.

SAP Object

The SAP (Service Application Point) object represents the application of a service on a device, without involving an interface. (Many services in IP Service Activator are modeled to be implemented on interfaces. However, some services can be implemented on devices as well, such as the Layer3 VPN Site.)

The SAP object has the following object inheritance:

```
SAP.Object
```

SAP objects are also used when interface-less VRFs are configured. The SAP (Service Application Point) object behaves similarly to an interface in this context, and provides a way of accessing and manipulating the VRF in the object model.

SAP objects are not visible in the GUI, but are accessible through the integration manager. When using the GUI, the creation of a SAP object occurs when a PE device is attached to a site. The device object icon is displayed in the Access Points folder under

the Site to represent the SAP. However, no SAP objects are visible or accessible through the GUI.

You must perform explicit lifecycle management on the SAP objects you manipulate through the integration manager.

You cannot directly link a device into a VPN site using the OIM. Instead, you must create an SAP object under the Device and then link the SAP to the Site.

SAP objects acquire most Interface object attributes. Although they can be modified through the integration manager, attributes that are not relevant for SAP objects (e.g. attributes other than Name, Description and IsSAP) are ignored.

Figure 4–45 shows the SAP object diagram.

Figure 4–45 SAP Object Diagram

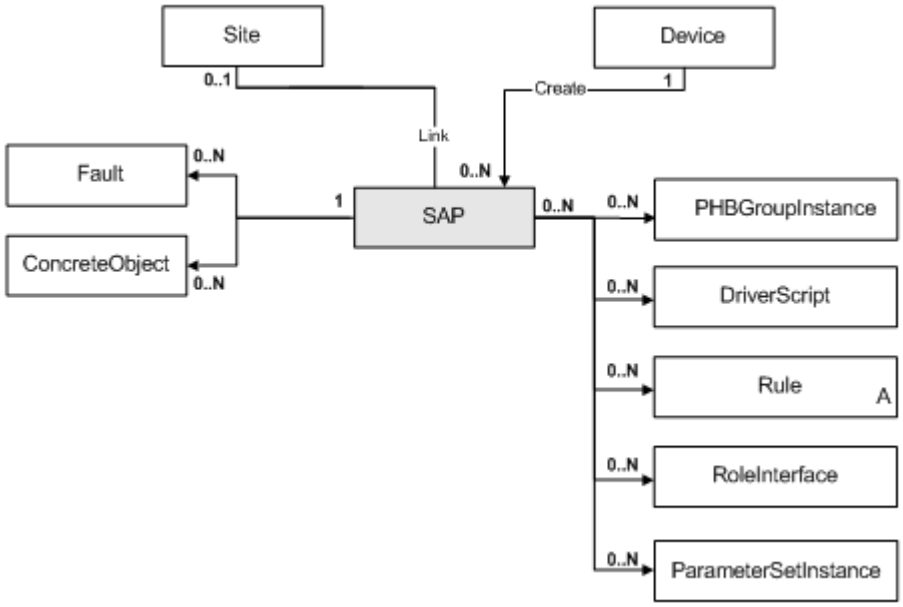


Table 4–99 describes the attributes for the SAP object.

Table 4–99 SAP Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RC	Name of SAP.
Description	String	""	RC	SNMP ifDescription.
IsSAP	Boolean	True	R	True if this is a Service Application Point (SAP) object.

Note: The remainder of the fields for the SAP object are identical to those of the Interface object. Many of these, however, have no meaning in the context of the SAP, even though they can be set through the Integration Manager.

Creation and Deletion

SAP objects must be created under the appropriate device. At creation time, the SAP requires a **Name**. An optional **Description** can also be provided. A read-only boolean attribute **IsSAP** will be set to True (boolean) to indicate that the object is an SAP.

Unlike the GUI created SAPs, integration manager created SAPs will have user-specified names.

Removal/deletion of the SAP object must be performed manually. A SAP object must be unlinked from all the services in which it participates. It can then be deleted in the context of the parent device.

Linking and Unlinking

Once a SAP object has been created, it can be linked into appropriate services (such as a VPN Site) in the same manner as an Interface object. SAP objects can also be parents to **RoleInterface** objects. A SAP object must have a child RoleInterface of type **Access** in order to participate in a VPN service.

To Create an Interface-less VRF

To create an interface-less VRF using the OIM, follow these steps:

- Create the target VPN Site object under the appropriate customer
- Create a SAP object under the target device
- Link a RoleInterface of **Access** to the SAP object
- Link the SAP to the VPN site
- Modify the SAP attributes as required (e.g. static route redistribution)
- Modify the Site attributes as required (e.g. turn on static routing, specify the VRF name)
- Link the Site into the VPN

SubInterface Object

A SubInterface object represents a sub-interface on a device interface. One object exists for each sub-interface. Sub-interfaces are always linked to the parent interface. These objects are created by reading the interface table of each device.

The SubInterface object has the following object inheritance:

```
SubInterface.Interface.Object
```

See Figure 4–44, "Interface Object Diagram" for the SubInterface object diagram.

The attributes for the SubInterface object are inherited from the Interface object. See Table 4–92, " Interface Object Attributes".

CreationMarkerSubInt Object

A CreationMarkerSubInt object represents a with a PVC. When this object is created, a SubInterface and a VCEndPoint are created on the relevant device. When the object is deleted, the relevant interface gets removed.

The CreationMarkerSubInt object has the following object inheritance:

```
SubInterface.CreationMarkerSubInt.Object
```

Figure 4–46 shows the CreationMarkerSubInt object diagram.

Figure 4–46 CreationMarkerSubInt Object Diagram

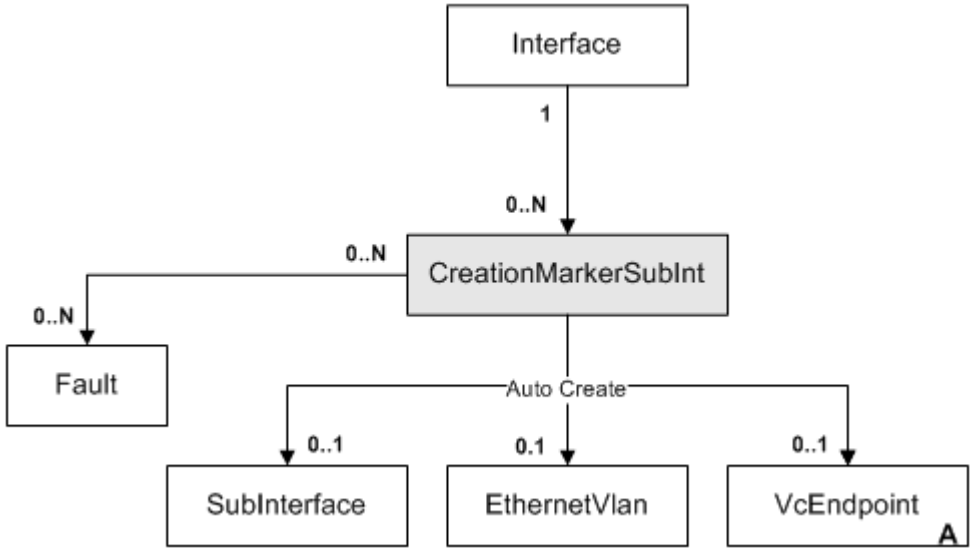


Table 4–100 describes the attributes for the CreationMarkerSubInt object.

Table 4–100 CreationMarkerSubInt Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RC	Name
SubInterfaceNumber	U32	0	RW	SubInterface number to be created.
Encapsulation	Enum	AtmAal5	RC	Encapsulation of the PVC created: 1 = AtmAal5 2 = AtmCell 3 = Frame 4 = Vlan
CreationState	Enum	Inactive	RO	State of the SubInterface and PVC creation. This is comparable to the state of a concrete object. 0 = Inactive 1 = Active 2 = Installed 3 = Failed
Conflict	Boolean	False	RO	Indicates whether this object is in conflict with another policy element.

CreationMarkerVcFr Object

A CreationMarkerVcFr object represents a Frame Relay VC endpoint, which is created directly under the interface. This can only be applied to some devices, depending on their capabilities.

The CreationMarkerVcFr object has the following object inheritance:

```
SubInterface.CreationMarkerVcFr.Object
```

Figure 4–47 shows the CreationMarkerVcFr object diagram.

Figure 4–47 CreationMarkerVcFr Object Diagram

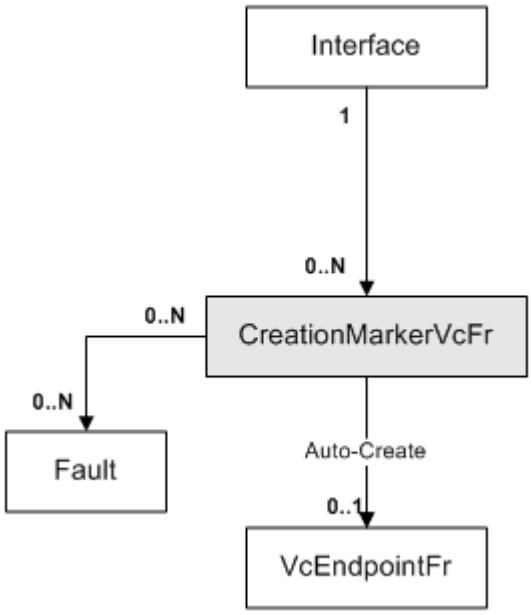


Table 4–101 describes the attributes for the CreationMarkerVcFr object.

Table 4–101 CreationMarkerVcFr Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RC	Simple accessory to this object.
CreationState	Enum	Inactive	RO	State of the SubInterface and PVC creation. This is comparable to the state of a concrete object. 0 = Inactive 1 = Active 2 = Installed 3 = Failed
Conflict	Boolean	False	RO	Indicates whether this object is in conflict with another policy element.

SubLayer Object

A SubLayer object represents one protocol sublayer of an interface, as returned by the device during the discovery process. This is especially used for AAL5 sublayers on ATM interfaces.

Figure 4–48 shows the SubLayer object diagram.

Figure 4–48 SubLayer Object Diagram

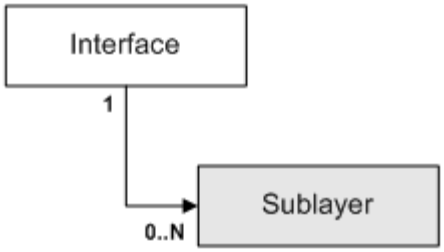


Table 4–102 describes the attributes for the SubLayer object.

Table 4–102 SubLayer Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of sublayer (SNMP ifDescr).
Number	U32	0	RO	Number of sublayer (SNMP ifIndex).
Speed	U32	0	RO	Speed of the sublayer in Kbits/s (SNMP ifSpeed parameter).
Type	U32	0	RO	Type of layer (ifType). This follows the IA[no default] ifType numbering.

VlanInterface Object

A VlanInterface object represents a VLAN interface. One object is created for each VLAN present on a device when the device is discovered. All relevant interfaces, sub-interfaces and segments are linked to it. Note that although the object can have a Role, you cannot apply any policy element to it.

The VlandInterface object has the following object inheritance:

```
VlanInterface.Object
```

Figure 4–49 shows the VlanInterface object diagram.

Figure 4–49 VlanInterface Object Diagram

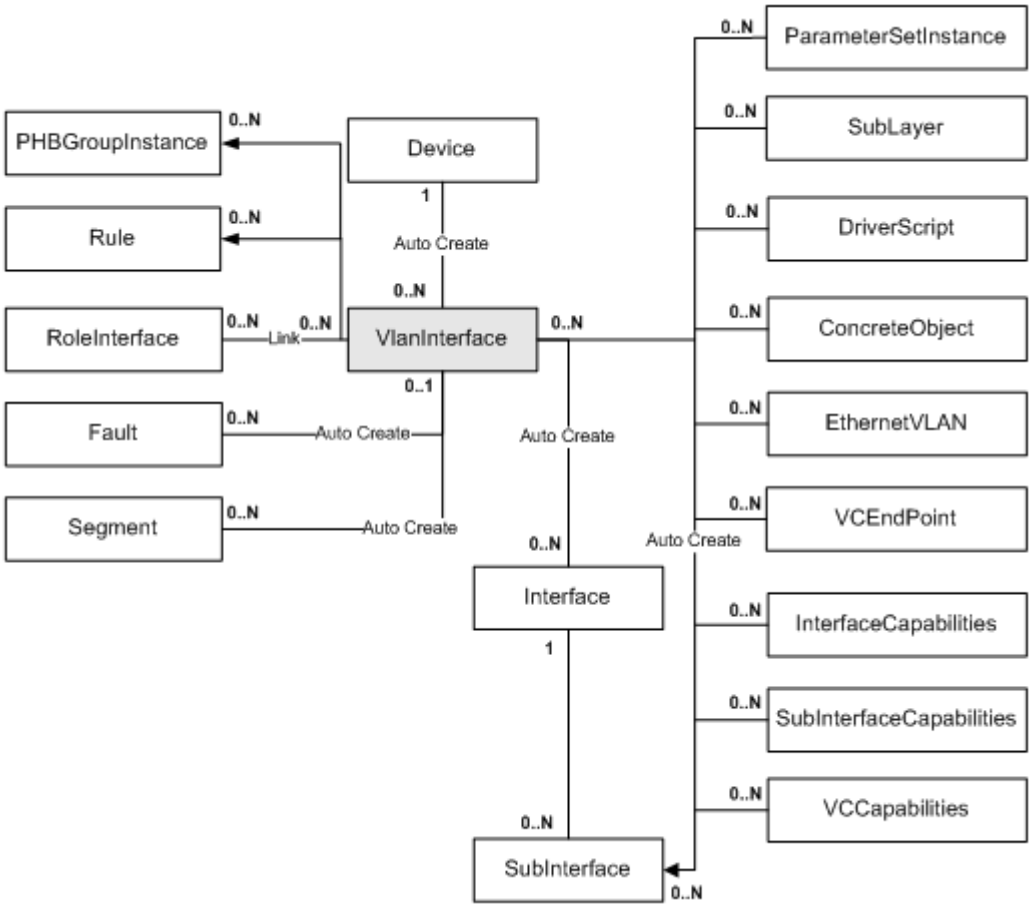


Table 4–103 describes the attributes for the VlanInterface object.

Table 4–103 VlanInterface Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RC	Name of VLAN interface.
Description	String	""	RC	SNMP ifDescription.
Number	U32	0	RC	Number of interface (SNMP ifIndex).
Type	U32	1	RC	SNMP ifType.
IpAddr	IPAddress	0.0.0.0	RC	IP Address of the VLAN interface.
SubnetMask	IPAddress	0.0.0.0	RC	Subnet mask.
PhysicalAddress	String	""	RC	Physical address of the interface (the MAC address).
Speed	U32	0	RC	Speed of the interface in Kbits/s (SNMP ifSpeed parameter).
VLANId	U32	0	RC	Vlan Number. Range: 0–4096

VcEndpoint Objects

This section describes the VcEndpoint objects.

VcEndpoint Object (Abstract)

The VcEndpoint abstract object maintains attributes common to all virtual circuit endpoints.

The abstract VcEndpoint object has the following object inheritance:

```
VcEndpoint.Object
```

Figure 4–50 shows the VcEndpoint object diagram.

Figure 4–50 VcEndpoint Object Diagram

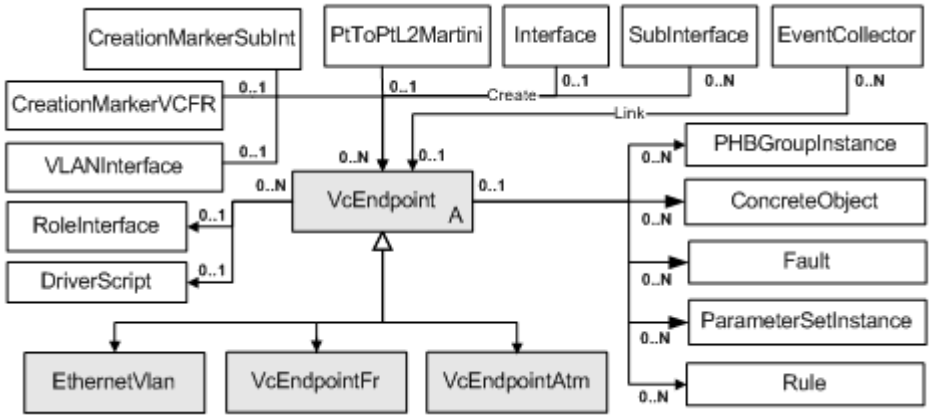


Table 4–104 describes the attributes for the VcEndpoint object.

Table 4–104 VcEndpoint Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RC	Name of VC endpoint. For ATM constructed from VPI and VCI. For Frame Relay constructed from DCLI.
Context	String	""	RW	Local context for driver scripts applied to VC endpoints (max 512 bytes).
EffectiveCommandDeliveryMode (This value is inherited when the command delivery mode is set at the Device, parent Interface, or parent SubInterface level.)	Enum	0	RO	0 = Online: commands delivered to device 1 = OfflineMaintenance: commands not delivered, concretes and other states updated 2 = OfflineTest: commands not delivered.

VcEndpointFr Object

VcEndpointFr objects represent Frame Relay virtual circuit endpoints.

The VcEndpointFr object has the following object inheritance:

```
VcEndpointFr.VcEndpoint.Object
```

See Figure 4–50, "VcEndpoint Object Diagram" for the VcEndpointFr object diagram.

Table 4–105 describes the attributes for the VcEndpointFr object.

Table 4–105 VcEndpointFr Object Attributes

Attribute Name	Type	Default	Access	Explanation
DLCI	U32	18	RC	Data Link Connection Identifier.
CircuitState	Enum	Inactive	RCW	State of the circuit: 1 = Invalid 2 = Active 3 = Inactive
CircuitType	Enum	PVC	RCW	Type of circuit: 0 = Unknown (Should never happen) 1 = PVC 2 = SVC (IP Service Activator only configures PVCs)
Context	String	""	RCW	Local context for driver scripts applied to VC endpoints (max 512 bytes).
EffectiveCommandDeliveryMode (This value is inherited when the command delivery mode is set at the Device, parent Interface, or parent SubInterface level.)	U32	0	RCW	0 = Online: commands delivered to device 1 = OfflineMaintenance: commands not delivered, concretes and other states updated 2 = OfflineTest: commands not delivered.
Name	String	"DLCI: 18"	RC	Name of VcEndpointFr object.
ID	U32	11902	RO	

VcEndpointAtm Object

VcEndpointAtm objects represent Asynchronous Transfer Mode (ATM) virtual circuit endpoints.

The VcEndpointAtm object has the following object inheritance:

```
VcEndpointAtm.VcEndpoint.Object
```

See Figure 4–50, "VcEndpoint Object Diagram" for the VcEndpointAtm object diagram.

Table 4–106 describes the attributes for the VcEndpointAtm object.

Table 4–106 VcEndpointAtm Object Attributes

Attribute Name	Type	Default	Access	Explanation
Vpi	U32	0	RC	Virtual Path Identifier.
Vci	U32	0	RC	Virtual Channel Identifier.
AdminStatus	Enum	3	RC	Desired administrative status of VC: 0 = Invalid (Should never happen) 1 = Up 2 = Down 3 = Unknown
OperStatus	Enum	3	RC	Current operational status of VC: 0 = Invalid (Should never happen) 1 = Up 2 = Down 3 = Unknown

EthernetVlan Object

An EthernetVlan object represents a virtual circuit endpoint on an Ethernet virtual local area network (VLAN).

The EthernetVlan object has the following object inheritance:

```
EthernetVlan.VcEndpoint.Object
```

See Figure 4–50, "VcEndpoint Object Diagram" for the EthernetVlan object diagram.

Table 4–107 describes the attributes for the EthernetVlan object.

Table 4–107 EthernetVlan Object Attributes

Attribute Name	Type	Default	Access	Explanation
VlanId	U32	0	RC	Vlan Number. Range: 1–4096
State	Enum	4 = Unknown	RC	0 = Down 1 = Up 2 = Testing 3 = Shutdown 4 = Unknown 5 = Dormant 6 = NotPresent 7 = LowerLayerDown 8 = NotFound 9 = NotYetDiscovered

Table 4–107 (Cont.) EthernetVlan Object Attributes

Attribute Name	Type	Default	Access	Explanation
Context	String	""	RCW	Local context for driver scripts applied to VC endpoints (max 512 bytes).
EffectiveCommandDeliveryMode (This value is inherited when the command delivery mode is set at the Device, parent Interface, or parent SubInterface level.)	U32	0	RCW	0 = Online: commands delivered to device 1 = OfflineMaintenance: commands not delivered, concretes and other states updated 2 = OfflineTest: commands not delivered.
Name	String	VLAN: 25	RC	Name of EthernetVlan object. For EthernetVlan constructed from VPI and VCI.
ID	U32	11854	RO	

Segment Object

For each interface which is not a point-to-point connection there will be one segment object representing the locally connected network segment.

The Segment object has the following object inheritance:

```
Segment.Object
```

Figure 4–51 shows the Segment object diagram.

Figure 4–51 Segment Object Diagram

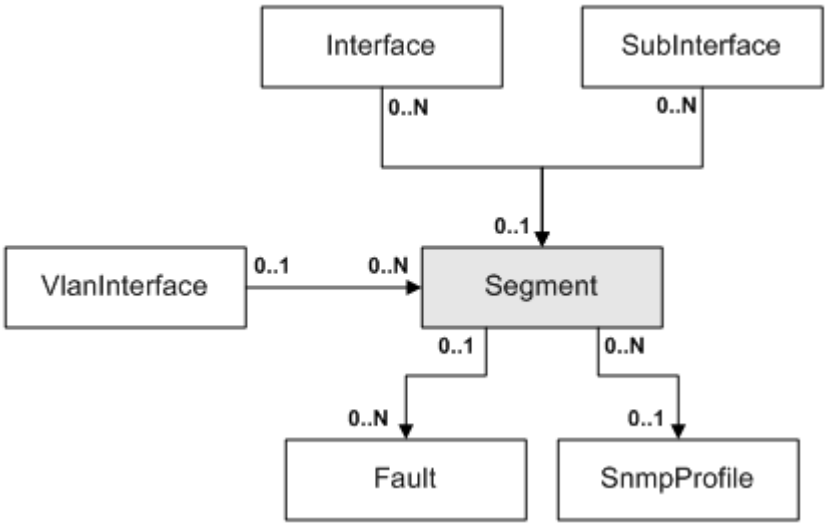


Table 4–108 describes the attributes for the Segment object.

Table 4–108 Segment Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	""	RO	Name of segment.
SubnetAddr	IPAddress	0.0.0.0	RC	Subnet address.
SubnetMask	IPAddress	0.0.0.0	RC	Subnet mask.
Community	String	""	RC	SNMP Read Community string used when running autodiscovery on this segment.
SnmpRetries	U32	2	RCW	The number of retry attempts when sending a PDU (Protocol Data Unit), Range: 0–20
SnmpTimeout	U32	3	RCW	Timeout of PDU response. Range: 1–30 seconds
Type	Enum	0	RC	Type of network segment: 0 = Unknown 1 = Other 2 = Serial 3 = BusSegment 4 = StarSegment 5 = TokenRing 6 = FddiRing 7 = AtmCloud 8 = FrCloud
Discovered	Boolean	""	RCW	True for segments created during discovery. False for segments created by the user, such as those linking a Virtual CE interface to the appropriate PE interface.
MaxRepetitions	U32	100	RW	Specifies the maximum number of rows that will fetch from a network resource in a single request when SNMPv2 discovery is used. The default value of this field is 100. Permitted range for this filed is 1 to 100. Decrease the value of Max-Repetitions if the routers don't respond to the default value of 100.

The System Model

This section describes the system model.

System Object

The System object represents the system root of the EOM. This object exists above all system objects.

The System object has the following object inheritance:

```
System.Object
```

Figure 4–52 System Object Diagram

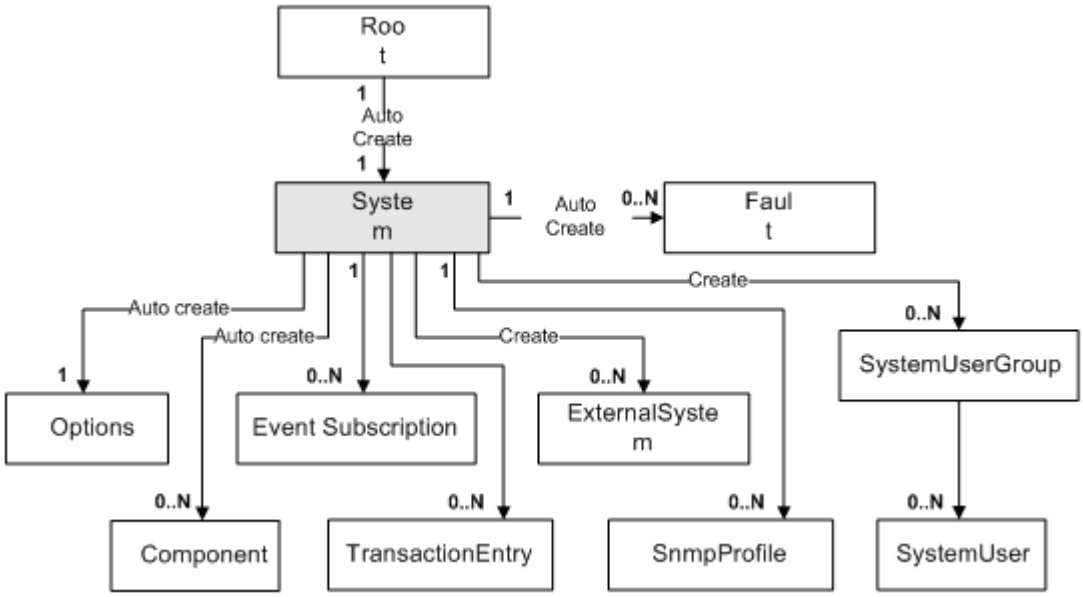


Table 4–109 describes the attributes for the System object.

Table 4–109 System Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	"System"	RO	Always "System".
ProxyPushState	Enum	0	RO	The current state of the push to the proxy agents (not the state of the proxy): 0 = Ready 1 = Push 2 = Pushing 3 = Invalid
ProxyPushProgress	U32	100	RO	When a proxy push is in progress, shows the percentage of the push that is complete.

Options Object

The Options object represents system-wide global options.

The Options object has the following object inheritance:

```
Options.Object
```

Figure 4–53 shows the Options object diagram.

Figure 4–53 Options Object Diagram

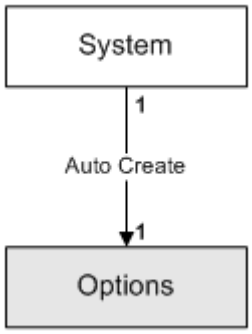


Table 4–110 describes the attributes for the Options object.

Table 4–110 Options Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	"Options"	RO	Always "Options".
MajorVersion	U32	3	RO	Object Model Major version.
MinorVersion	U32	0	RO	Object Model Minor version.
ProxyAssign	Enum	1	RW	Global option defining the strategy for assigning devices to Proxy Agents automatically: 0 = Off 1 = AssignFirst 2 = LoadBalance
Revision	U32	0	RO	Object Model Revision Version
UnmanagedAction	Enum	1	RW	Default behavior to apply when unmanaging a device, unless overridden at device level: 0 = RemoveConfiguration 1 = LeaveConfiguration
TransactionLimit	U32	200	RO	Number of applied transactions to keep.

TransactionEntry Object

A TransactionEntry object represents each queued transaction, allowing the user to see a list of these transactions and to execute or roll back a transaction.

The TransactionEntry object has the following object inheritance:

```
TransactionEntry.Object
```

A TransactionEntry object can be created with the **commit** command, and manipulated with the **schedule**, **rollback** and **merge** commands.

Figure 4–54 shows the TransactionEntry object diagram.

Figure 4–54 TransactionEntry Object Diagram

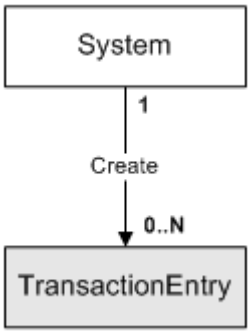


Table 4–111 describes the attributes for the TransactionEntry object.

Table 4–111 TransactionEntry Object Attributes

Attribute	Type	Default	Access	Explanation
Name	String	[no default]	RW	Unique name for transaction.
Description	String	[no default]	RW	Description of transaction entered by user, entered as a summary to describe the transaction.
Username	String	Name of user logged in	RO	Username of the user that created this transaction. Left empty if applicable to stored automatically-generated system transactions.
RollbackOnFailure	Boolean	True	RC	True unmerges (rolls back) the transaction if it fails. False does not roll back the transaction.
ReasonForFailure	String	[no default]	RW	This attribute contains the messages from any faults raised for this transaction if the transaction monitor is running.
State	Enum	0	RC	Status of this object: 0 = Pending 1 = Installed 2 = Uninstalled 3 = Scheduled 4 = ScheduledUninstall 5 = FailedScheduled 6 = FailedScheduleUninstall 7 = Committed 8 = ScheduleCommitted 9 = SucceededScheduleUninstall

Table 4–111 (Cont.) TransactionEntry Object Attributes

Attribute	Type	Default	Access	Explanation
Provisioning Status	Enum	[no default]	RW	Provisioning status of this object: 0 = Pending: initial state on commit of the transaction. 1 = Succeeded: all concretes are in the Installed state or deleted. 2 = Failed: at least one concrete is in Rejected state, or the transaction timed out and the transaction monitor is configured to fail transactions on timeout. 3 = Timedout: the maximum time a transaction is allowed to stay in pending state awaiting completion notifications was exceeded. More details on these states are provided in *IP Service Activator System Administrator's Guide*.
Schedule	DateTime	Now	RW	Date and Time to schedule this transaction for or when transaction was changed to current static state.
Operations	String	[no default]	RC	The data appearing below the data operations field is a deserialized representation of objects stored in a string type within the TransactionEntry object. The fields available for viewing represent the following: CID = ID of the concrete operated on Status = Inactive, Pending, Succeeded, Failed, or Timedout Operation = Type of operation (Create, Delete or Link) PClass = Parent class PName = Parent name ParentID = ID of the parent object for the operation
NumberofConcretes	U32	[no default]	R	The number of concretes for the transaction.

Component Object

A Component object represents an IP Service Activator component: Component Manager, Policy Server, Event Handler, OIM, Proxy Agent, Device Driver, System Logger or Integration Component. The Type attribute identifies the type of component; individual components are not accessible via the EOM.

The Component object has the following object inheritance:

```
Component.Object
```

Figure 4–55 shows the Component object diagram.

Figure 4–55 Component Object Diagram

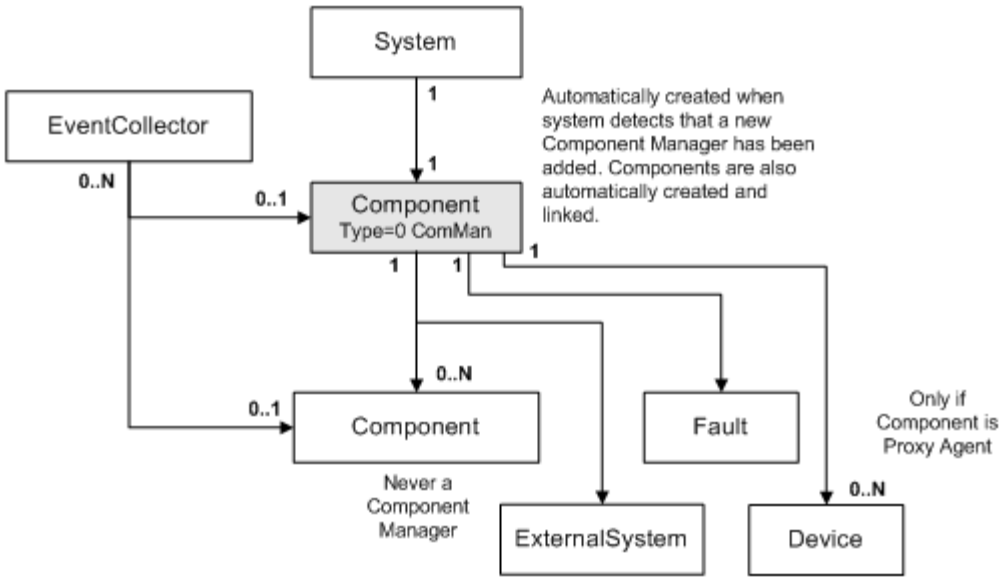


Table 4–112 describes the attributes for the Component object.

Table 4–112 Component Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of component.
Remarks	String	[no default]	RO	Optional additional comments. Limit of 255 characters.
CorbaName	String	Auto	RO	CORBA name of component.
CorbaKind	String	Auto	RO	CORBA type of component.
CorbaLocation	String	Auto	RO	CORBA location of component.
CorbaMajor	U32	0	RO	Current version of the CORBA interface - major number.
CorbaMinor	U32	0	RO	Current version of the CORBA interface - minor number.
RunState	Enum	0	RO	Allows triggering shutdown of components on a single host. Can be one of: 0-2 = running 3-5 = shutdown
StartTime	U32	[no default]	RO	Time since this component started, in seconds.
Restarts	U32	0	RO	The number of times this component has been restarted.

Table 4–112 (Cont.) Component Object Attributes

Attribute Name	Type	Default	Access	Explanation
Type	String	[no default]	RO	Identifies type of component; one of: "ComMan" "PolicyServer" "ProxyAgent" "Driver" "SysLog" "ServiceComponent" "OssIntegrationManager" "EventHandler" "IntegrationComponent"
UpTime	String	0	RO	Elapsed time since the component started, specifying days, hours, minutes and seconds.
IsNetworkProcessor	Boolean	False	RO	Indicates if the proxy is a Network Processor or a regular proxy agent.
Id	U32	[no default]	RO	Unique ID used to reference this object.
Full Version	String	[no default]	RO	The FullVersion attribute contains the values for the MajorVersion, MinorVersion, ServicePack and BuildNo.

EventSubscription Object

An Event Subscription represents the definition of an external system subscribing to events and faults reported by IP Service Activator. The subscription defines the delivery type such as via an SNMP trap, NetCool, CORBA interface, etc.

The EventSubscription object has the following object inheritance:

```
EventSubscription.Object
```

Figure 4–56 shows the EventSubscription object diagram.

Figure 4–56 EventSubscription Object Diagram

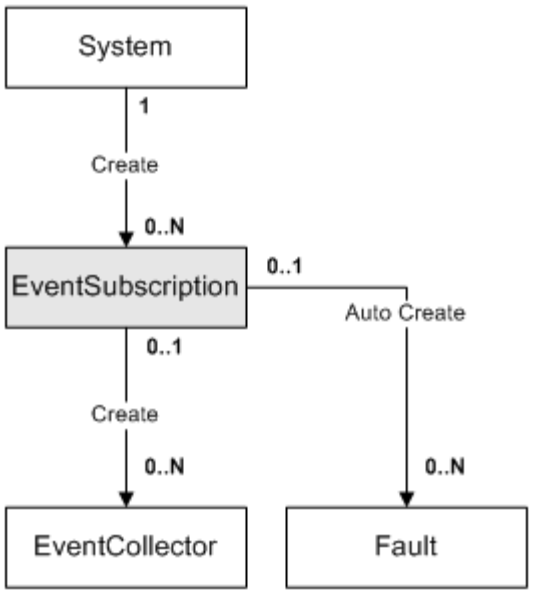


Table 4–113 describes the attributes for the EventSubscription object.

Table 4–113 EventSubscription Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	"InfoVista "	RCW	Name of subscription.
Description	String	[no default]	RCW	A longer description.
Enable	Boolean	True	RCW	True = subscription enabled. False = subscription disabled.
SendPendingEvents	Boolean	False	RCW	True = currently pending events are to be delivered. False = pending events are not delivered.
DeliveryType	U32	4	RCW	Delivery method: 0=SnmpTrap 1=UpgradedSnmpTrap 2=Netcool 3=CorbaChannel 4=DatabaseOnly

Table 4–113 (Cont.) EventSubscription Object Attributes

Attribute Name	Type	Default	Access	Explanation
DeliveryDetails	String	[no default]	RCW	String formatted according to type. For SNMP, a comma-separated list of IP address, port and version, for example, "192.168.1.2,42,2". For CORBA and Netcool, the name of service. For DatabaseOnly: blank.
ImpactTargets	U32	0	RCW	Bitwise value, 1 bit per impact target: Bit 0 = Customer, Site, VPN Bit 1 = Device Bit 2 = Interface
HighestSeverity	Boolean	False	RCW	True = delivers highest severity faults only. False = all faults delivered.
TransactionEvents	U32	0	RCW	Identifies transaction start/end events to be reported: 0 = None 1 = User 2 = System 3 = All

EventCollector object

Event Collector objects are linked to other objects in the EOM to specify that the object is to be monitored.

The EventCollector object has the following object inheritance:

```
Subscription.Object
```

To set the collection point to a particular object then you should link the object to the EventCollector. If the collection point is the Root object, then no object should be linked, but the RootCollection attribute should be set to True. If the collection point is the Policy or System objects, then no object should be linked, but the CollectionPoint attribute should be set to 1 or 2 respectively.

Figure 4–57 shows the EventCollector object diagram.

Figure 4–57 EventCollector Object Diagram

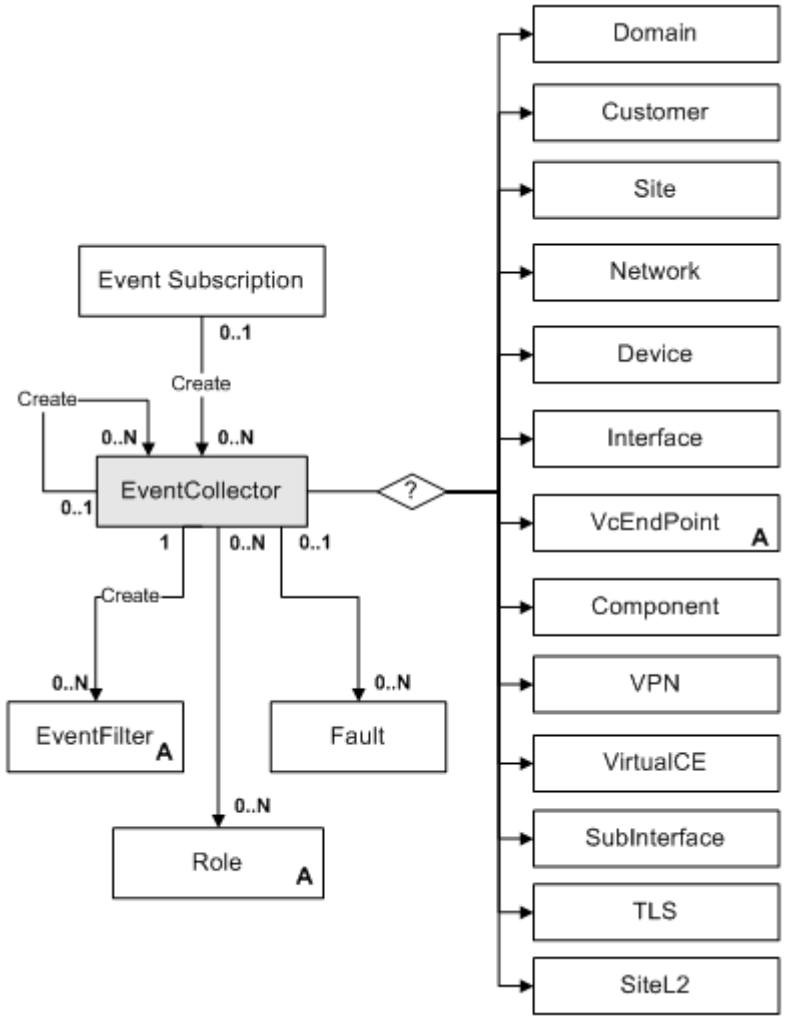


Table 4–114 describes the attributes for the EventCollector object.

Table 4–114 EventCollector Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of collector, must be unique.
Description	String	[no default]	RW	Remarks.
CollectEvents	Boolean	True	RW	True = events are collected. False = events are not collected.
CollectionPoint	U32	0	RW	Used if the collection point is the Policy (1) or System (2), as these objects cannot be linked directly.

Table 4–114 (Cont.) EventCollector Object Attributes

Attribute Name	Type	Default	Access	Explanation
RootCollection	Boolean	False	RW	True = Area of interest is the entire object model. False = Area of interest is defined by CollectionPoint.
HierarchicalCollection	Boolean	True	RW	True = collection is expanded below the local scope from the CollectionPoint. False = collection affects CollectionPoint only.
Type	Enum	1	RW	Specifies the relevant Event Type: 0 = CreateAndDelete 1 = Fault 2 = AttributeChange 3 = StateChange 4 = Link/Unlink
Classname	String	[no default]	RW	Name of class being subscribed to. Must be a valid class name. Blank if subscription relates to a specific object.

EventFilter Objects

This section describes the EventFilter objects.

EventFilter Object (Abstract)

The EventFilter object is used by the Event Collector to determine which events are to be generated. It operates on event specific details such as the attribute that has changed on an object described in the event, or the fault ID raised.

The abstract EventFilter object has the following object inheritance:

```
EventFilter.Object
```

Figure 4–58 shows the EventFilter object diagram.

Figure 4–58 EventFilter Object Diagram

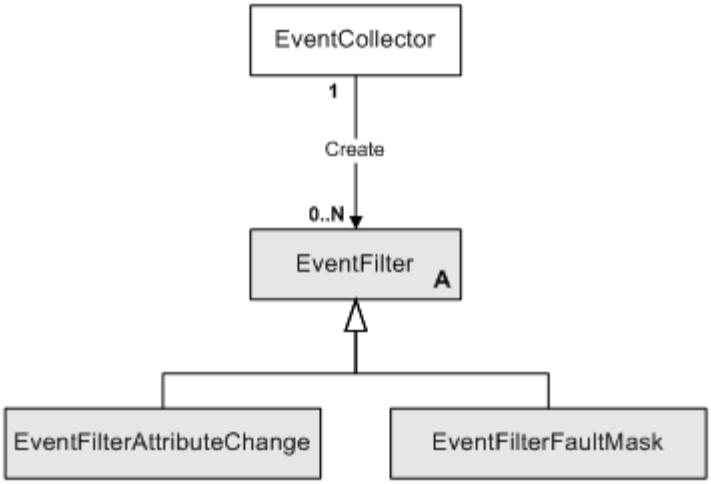


Table 4–115 describes the attributes for the EventFilter object.

Table 4–115 EventFilter Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	R	Name of event filter
Permit	Boolean	True	RCW	False=Deny matching events. True=Permit matching events.
Priority	U32	0	RCW	Ordering of filter.

EventFilterAttributeChange object

The EventFilterAttributeChange object performs a match between an attribute name and the changes on an object to decide whether the event should be generated. The name corresponds to the type of filter.

The EventFilterAttributeChange object has the following object inheritance:

```
EventFilterAttributeChange.EventFilter.Object
```

See Figure 4–58 for the EventFilterAttributeChange object diagram.

Table 4–116 describes the attribute for the EventFilterAttributeChange object.

Table 4–116 EventFilterAttributeChange Object Attributes

Attribute Name	Type	Default	Access	Explanation
AttributeName	String	[no default]	RCW	String to match against attribute name.

EventFilterFaultMask Object

The EventFilterFaultMask object performs a match between a triggered event and details of the faults to be reported. The name corresponds to the type of filter.

The EventFilterFaultMask object has the following object inheritance:

```
EventFilterFaultMask.EventFilter.Object
```

See Figure 4–58 for the EventFilterFaultMask object diagram.

Table 4–117 describes the attributes for the EventFilterFaultMask object.

Table 4–117 EventFilterFaultMask Object Attributes

Attribute Name	Type	Default	Access	Explanation
FaultCode	U32	0	RCW	Fault code; only used if FaultCategory is set to SingleFaultCode.
FaultCategory	U32	0	RCW	Category of faults to be filtered: 0=SingleFaultCode 1=ComponentFaultClears 2=DatabaseAccessFaultClears 3=ProvisioningFaultClears 4=PolicyRuleFaultClears 5=CommunicationFaultClears 6=DeviceOrInterfaceFaultClears 7=UpgradedComponentFaults 8=UpgradedDatabaseAccessFaults 9=UpgradedPolicyOrServiceFaults 10=UpgradedDeviceFaults 11=UpgradedInterfaceFaults 12=UpgradedLinkFaults 13=UpgradedSystemFaults 14=UpgradedConfigurationFaults

ExternalSystem Object

The ExternalSystem object represents an external system, and can be linked to a component of type IntegrationComponent.

Figure 4–59 shows the ExternalSystem object diagram.

Figure 4–59 ExternalSystem Object Diagram

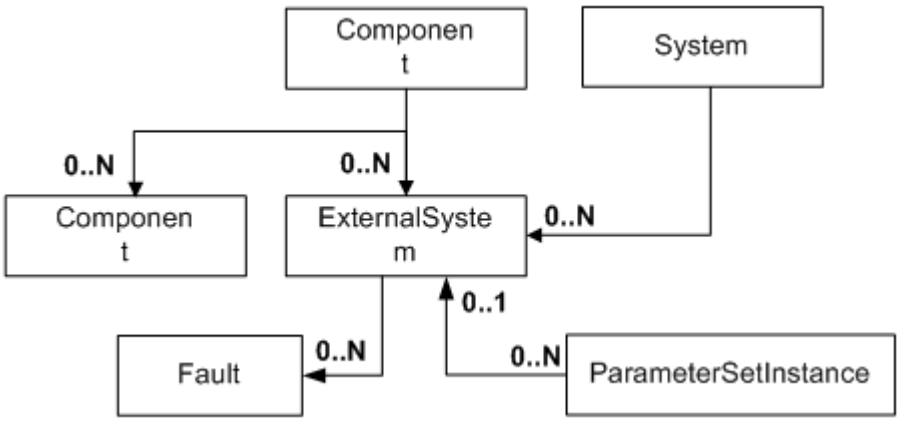


Table 4–118 describes the attributes for the ExternalSystem object.

Table 4–118 ExternalSystem Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name of the specific instance of external system.
Remarks	String	[no default]	RW	Optional additional comments. Limit of 255 characters.
Type	String	[no default]	RW	Type of external system.
IpAddr1	IPAddress	0.0.0.0	RW	Primary IP address of external system.
IpAddr2	IPAddress	0.0.0.0	RW	Secondary IP address of external system.
Port1	U32	0	RW	Primary Port number.
Port2	U32	0	RW	Secondary Port number.
UserName	String	[no default]	WO	User login name (encrypted).
Password	String	[no default]	WO	User password (encrypted).
URL	String	[no default]	RW	URL on which to contact external system.

Fault Object

A Fault object represents a system message indicating a fault reported from an IP Service Activator component or a network object. Fault objects exist while a component has an outstanding fault. When the fault is fixed the fault object is automatically removed. A fault object is linked to the object representing the component that currently has a fault.

The Fault object has the following object inheritance:

```
Fault.Object
```

Figure 4–60 shows the Fault object diagram.

Figure 4–60 Fault Object Diagram

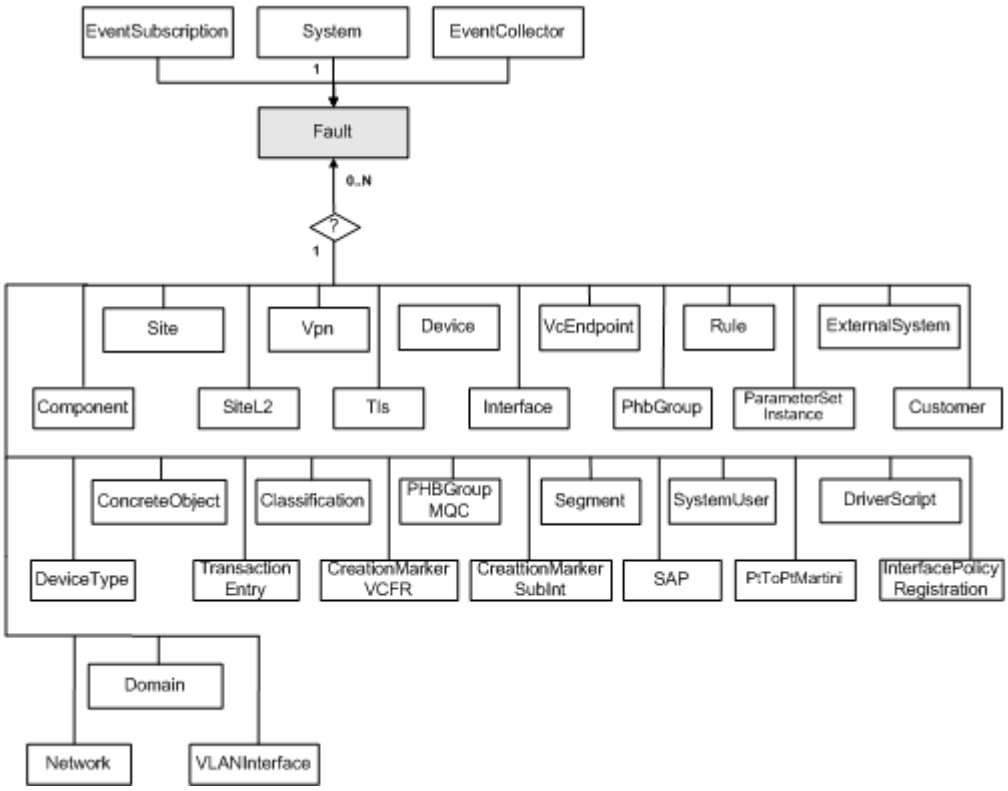


Table 4–119 describe the attributes for the Fault object.

Table 4–119 Fault Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RO	Name of the fault.
TimeStamp	DateTime	Now	RC	Time of fault creation.
MajorCode	U32	0	RC	Major error code (this is 100 for faults raised by IP Service Activator).
MinorCode	U32	0	RC	Minor error code.
Severity	Enum	0	RC	The severity level of the fault. One of the following: 0 = Info 1 = Notice 2 = Warning 3 = Error 4 = Critical
Error	String	[no default]	RO	Description text with parameter place holders.

SnmpProfile object

An SnmpProfile object represents a user-defined profile of SNMP attributes that can be assigned to manage the discovery of a group of devices, or to the re-discovery of individual devices.

The SnmpProfile object has the following object inheritance:

```
SnmpProfile.Object
```

Figure 4–61 shows the SnmpProfile object diagram.

Figure 4–61 SnmpProfile Object Diagram

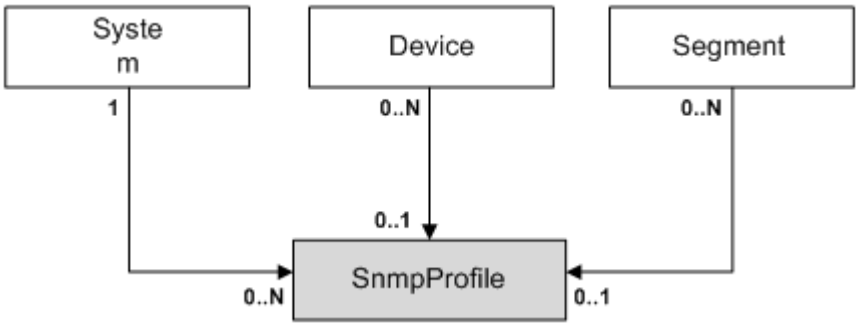


Table 4–120 describes the attributes for the SnmpProfile object.

Table 4–120 SnmpProfile Object Attributes

Attribute Name	Type	Default	Access	Explanation
Name	String	[no default]	RW	Name given to this SNMP Profile
Description	String	[no default]	RW	Text field describing this SNMP Profile
SnmpRetries	U32	2	RW	The number of retry attempts when sending a PDU (Protocol Data Unit). Range: 0–20
SnmpTimeout	U32	3	RW	Timeout of PDU response. Range: 1–30 seconds
SnmpVn	Enum	3	RW	Version of SNMP to use to interrogate the device: 0 = None 1 = SnmpV1 2 = SnmpV2c 3 = SnmpV1V2c 4 = SnmpV3 5 = SnmpV1V3 6 = SnmpV2cV3 7 = SnmpV1V2cV3

Table 4–120 (Cont.) SnmpProfile Object Attributes

Attribute Name	Type	Default	Access	Explanation
Community	Secure	public	RW	SNMP Community string used for Read access to network devices, when running autodiscovery
Userid	Secure	noAuthUser	RCW	UserID authorized to set authentication and privacy passwords
Authentication	Enum	None	RW	0 = None 1 = SHA 2 = MD5
AuthenticationPassword	String	[no default]	RW	Password length: 8-127 characters
Privacy	Enum	None	RW	0 = None 1 = DES 2 = AES
PrivacyPassword	String	[no default]	RW	Password length: 8-127 characters
Id	U32	541	RO	Unique ID used to reference this object
MaxRepetitions	U32	100	RW	Specifies the maximum number of rows that will fetch from a network resource in a single request when SNMP V2c is used for discovery. Permitted range for this filed is 1 to 100. Decrease the value of Max-Repetitions if the routers don't respond to the default value of 100.

SystemUserGroup Objects

This section describes the SystemUserGroup objects.

SystemUserGroup Object

The SystemUserGroup object is read only. The administrator must log into the GUI to create a system user group.

The SystemUserGroup object has the following object inheritance:

```
SystemUserGroup.Object
```

A system user group defines the access level that its members have within IP Service Activator. Every system user is a member of only one system user group. All users in the same system user group have the same access privileges.

A system user group can have one of the following access levels:

- Super User: users can access the user interface, view information and perform any operation. Only users with Super User access can set up other users and perform configuration tasks associated with the policy server.

- Read Write: users can access the user interface, view information, and create and/or modify objects. The operations that can be performed by group members depend on the permissions that are set for the group.
- Read Only: users can access the user interface and view objects but cannot create new or modify existing objects.

For system user groups with Read Write access, you can specify exactly which operations its members can perform. For example, you can specify that members cannot create new networks, but can discover new devices. These controls are referred to as permissions.

Figure 4–62 shows the SystemUserGroup object diagram.

Figure 4–62 SystemUserGroup Object Diagram

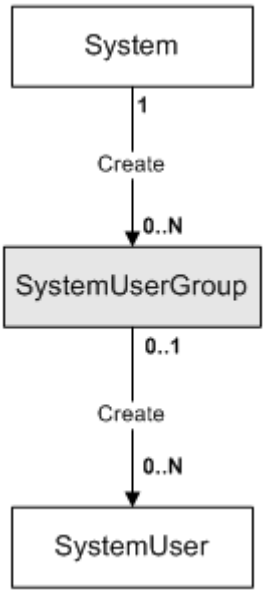


Table 4–121 describes the attributes for the SystemUserGroup object.

Table 4–121 SystemUserGroup Object Attributes

Attribute Name	Type	Default	Access	Explanation
AccessRights				None = 0, SystemWrite = 1, SystemRead = 2, WorldWrite = 4, WorldRead = 8, ReadOnly = 8, ReadWrite = 8 + 4, SuperUser = 8 + 4 + 2 + 1 Only a Super user sees System objects.
Name	String	[no default]	R	Name of the user group.
Id	U32	0	R	This is a unique ID used to reference this object.

SystemUser Object

The SystemUser object is used to create new users and set security restrictions. System users are set up within a system user group. All users in the same system user group have the same access privileges. A user can be a member of only one group.

The SystemUser object has the following object inheritance:

```
SystemUser.SystemUserGroup.Object
```

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

See Figure 4–62 for the SystemUser object diagram.

Table 4–122 describe the attributes for the SystemUser object.

Table 4–122 SystemUser Object Attributes

Attribute Name	Type	Default	Access	Explanation
Active	U32	0	R	This value indicates the number of concurrent sessions that the user is running. 0 indicates that the user is not currently logged on.
Concurrent	Boolean	False	RCW	False prevents more than one concurrent logon by this user
Enabled	Boolean	True	RCW	False disables user logon.
ExpireDate	Date	[date]	R	Password expiry date is calculated from the value entered for PasswordExpires. The default [date] is 30 days after the user is created.
FailedLogins	U32	0	R	The number of failed logon attempts before the user is denied access (locked out).
Name	Secure	none	RCW	Name of the user.
Password	String	[no default]	RCW	User password string.
PasswordExpires	U32	30	RCW	Number of days between password change and password expiry. A value of 0 means "password never expires". When login is blocked due to password expired, the user must log into the GUI and change the password. This will enable login to the OSS Integration Manager.
Remarks	String	[no default]	RCW	Add any desired text. Limit of 255 characters.

Table 4–122 (Cont.) SystemUser Object Attributes

Attribute Name	Type	Default	Access	Explanation
ResetPassword	Boolean	False	RCW	True requires the user to reset the password at initial logon. When this attribute is True (that is, when the password is expired), the user cannot log into the OSS Integration Manager. When login is blocked due to reset required by the system administrator, the user must log into the GUI and change the password. This will enable login to the OSS Integration Manager.
Id	U32	0	R	A unique reference number for each object.

5

Examples of Using OIM

This chapter provides examples of using the OIM commands to perform a number of functions within Oracle Communications IP Service Activator.

Explanation of the examples

This chapter consists of a number of worked examples showing some of the most common uses of OIM, from discovering devices to setting up management and customer VPNs and applying policy rules to them. The examples are based on the use of the OIM CLI but as they are intended as general illustrations, they can be adapted for use in scripts.

For illustrative purposes, example screens are included showing the corresponding appearance of the user interface.

The examples are based on a simple example network, initially with a single domain called "iisp".

Browsing the Topology

A set of commands are available to allow users to navigate through the system and display information.

Pre-requisites

None

Command syntax

The following commands allow you to navigate through the network topology. See "The Navigation Module" for full details.

Table 5–1 Command Syntax

Command	Purpose
getPath [*object-path*]	Returns the path to the current position in the EOM
setPath [*object-path*]	Sets the current path within the EOM
getChildren [*object-path*]	Lists the children of the object
getName [*object-path*]	Returns the name of an object
getparents [*object-path]*	Lists the parents of an object

Table 5–1 (Cont.) Command Syntax

Command	Purpose
getAttributes *[object-path]*	Lists the attributes of an object
getID [*object-path*]	Returns the ID of an object

Object Hierarchy

The following diagrams illustrate the basic structure of the object hierarchy for navigation purposes.

Figure 5–1 shows the Root object hierarchy. Note the standard ID numbers for these objects.

Figure 5–1 Root Object Hierarchy

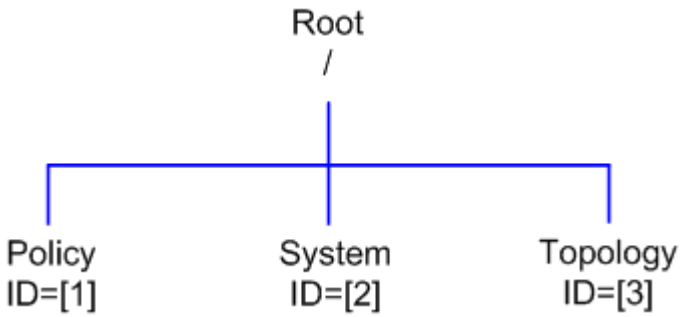


Figure 5–2 shows the Policy object hierarchy.

Figure 5–2 Policy Object Hierarchy

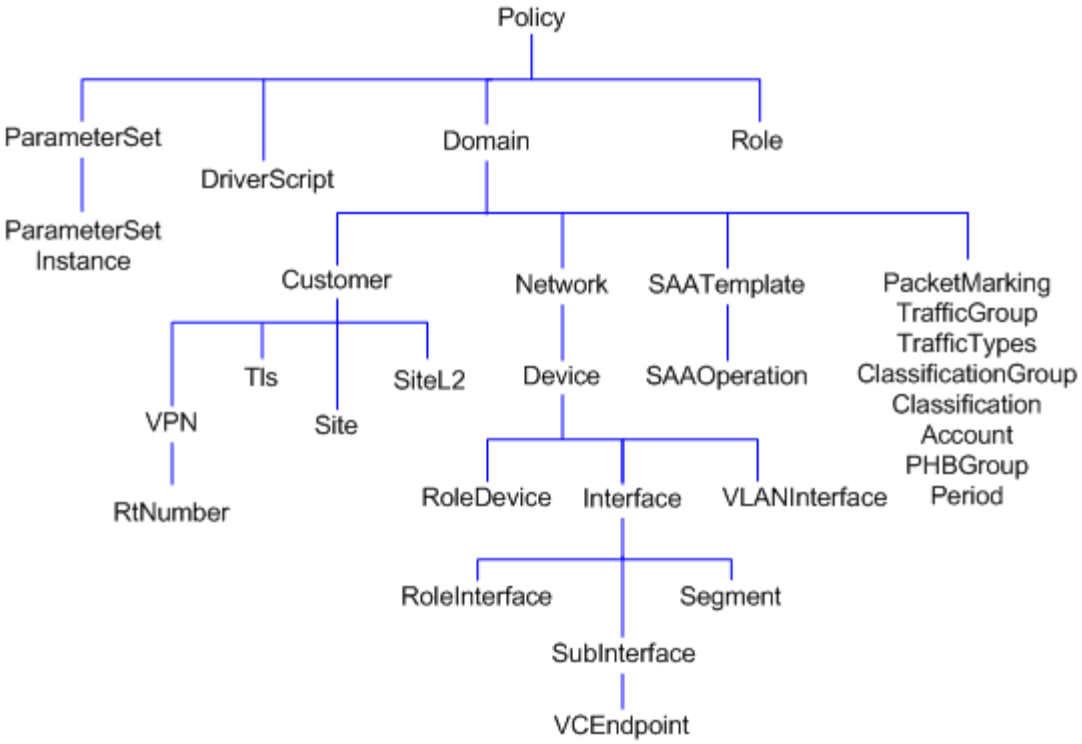


Figure 5–3 shows the System object hierarchy.

Figure 5–3 System Object Hierarchy

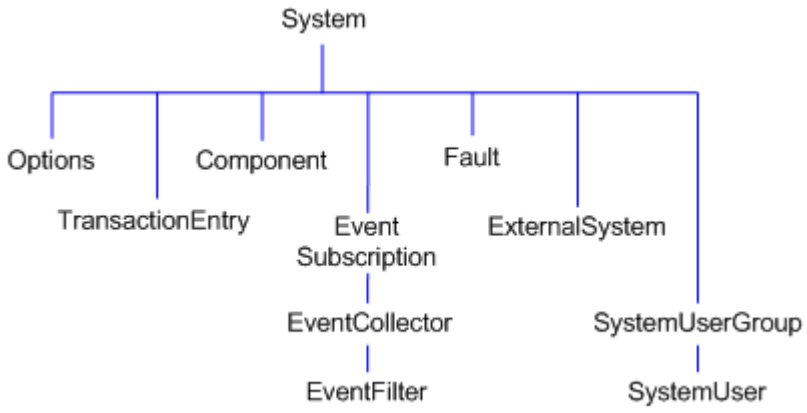
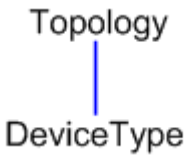


Figure 5–4 shows the Topology object hierarchy. Note that whenever further objects are created or existing objects linked they appear as children of the appropriate object. For example, if a rule is created at the device level it appears as a child of the device, or if a PE interface is linked to a site, it will appear as a child of the site.

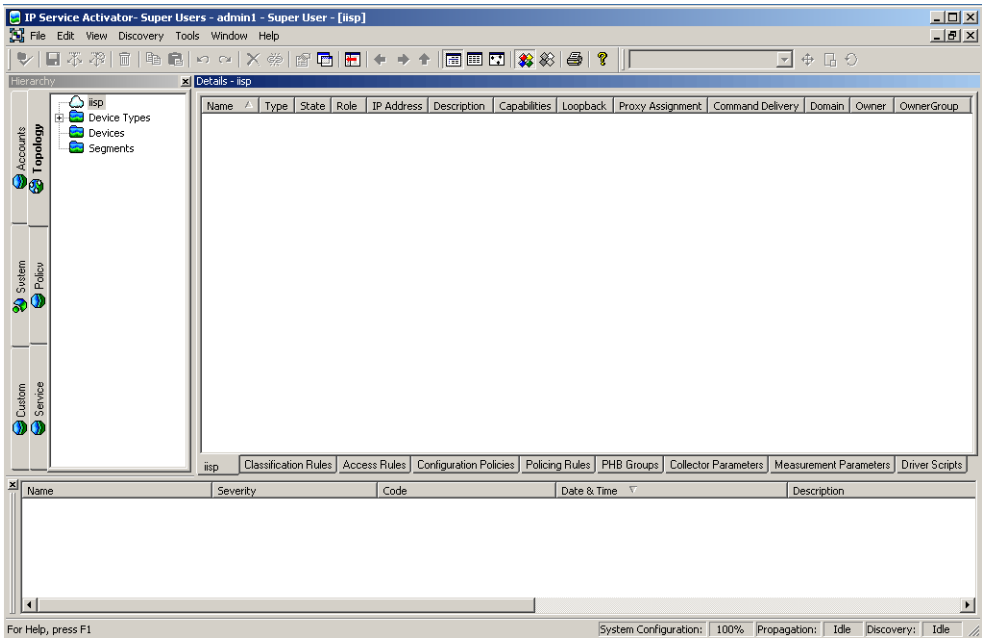
Figure 5–4 Topology Object Hierarchy



Example

The following examples show commands to navigate through the system and list the pre-existing objects. This assumes one domain called **iisp**, as shown in Figure 5–5.

Figure 5–5 Single iisp Domain



To browse the policy part of the EOM, set the path to "policy":

```
setPath policy:policy
    /Policy:"Policy"
/Policy:"Policy">getchildren
    [4] RoleDevice:"Access"
    [5] RoleDevice:"Gateway"
    [6] RoleDevice:"Core"
    [7] RoleDevice:"Any"
    [8] RoleInterface:"Core"
    [9] RoleInterface:"Customer"
    [10] RoleInterface:"Local"
    [11] RoleInterface:"Access"
    [12] RoleInterface:"Any"
    [272] Domain:"iisp"
```

You can then view the domain:

```
/Policy:"Policy">setPath domain:iisp
    /Policy:"Policy"/Domain:"iisp"
/Policy:"Policy"/Domain:"iisp">getchildren
```

```
    [273] Network:"OrchDemo"
    [285] PacketMarking:"IP Precedence 0"
    [286] PacketMarking:"IP Precedence 1"
    [287] PacketMarking:"IP Precedence 2"
    [288] PacketMarking:"IP Precedence 3"
    [289] PacketMarking:"IP Precedence 4"
    [290] PacketMarking:"IP Precedence 5"
    [291] PacketMarking:"IP Precedence 6"
    [292] PacketMarking:"IP Precedence 7"
    [293] TrafficGroup:"Standard Traffic Types"
    [297] TrafficGroup:"Standard Port Numbers"
    [647] ClassificationGroup:"Standard Traffic Types"
    [651] ClassificationGroup:"Standard Port Numbers"
    [1749] Customer:"Customer A"
    [1756] Customer:"Management"
```

You can then view the network and view the devices that have been set up:

```
/Policy:"Policy"/Domain:"iisp">setPath network:"OrchDemo"
    /Policy:"Policy"/Domain:"iisp"/Network:""

/Policy:"Policy"/Domain:"iisp"/Network:"">getChildren
    [1212] Device:"G3640-4.Orchestream.com"
    [1213] Device:"G3640-6.Orchestream.com"
    [1214] Device:"G3640-3.Orchestream.com"
    [1570] Network:"CORE NET"
    [1847] Device:"A2501-3.Orchestream.com"
```

Finding Objects

The find command allows you to search for a particular character string.

Pre-requisites

The objects must exist.

Command Syntax

The basic syntax of the **find** command is:

`find [`*`object-path`*`] `*`search-string attributes`*` finddirection=`*`direction`*` [depth=`*`depth`*`]`

object-path: The ID or path of the object from which to start the search. If this is omitted, the search will start from the current point in the path.

search-string: A string, enclosed in quotes, containing the object to look for. It may contain wild cards.

attributes: Any number of attributes may be specified. An object will only satisfy the search criteria if it matches the search string and has the same attributes as those given.

direction: Indicates the direction of the search (**child** or **parent**). By default the Find operation searches downwards through the hierarchy, from parent to child, but you can set it to search upward, from child to parent.

depth: Limits the number of levels in the object tree hierarchy that are included in a search. By default the Find operation executes a parallel search across and down all siblings.

Examples

To find all objects from the current location:

```
find . "*"
```

To search for a domain in a current directory:

```
find . Domain:"*"
```

To find a specific object from the current location:

```
find . Device:"DeviceName"
```

To search up the tree from the current location:

```
find . Customer:"*" finddirection=parent
```

To find sites that are immediate children of a particular customer:

```
find site:"*" Policy:"policy"/Domain:"domain"/Customer:"customer" depth=1
```

If you are unsure of where an object is in the object model, it is a good idea to start the search from the root. For example:

```
find / device:"G*"
[1152] Device:"G4500-1"
[1178] Device:"G4500-3"
[1225] Device:"G3640-3"
[1266] Device:"G7204-1"
[1817] Device:"GCAT5509"
[1837] Device:"G3640-7"
[1852] Device:"G-6509"
```

Discovering Devices

This example describes how to discover devices using OIM. It is possible to discover new devices, set security/authentication parameters and fetch the capabilities of interfaces using OIM in exactly the same way as from the user interface.

Note that the discovery can take some time especially if a large number of devices are discovered.

Pre-requisites

- A domain must previously have been created using the UI.
- The ID or name of the network object which will be the parent of the discovered devices must be known.

Command Syntax

The basic syntax for the **discover** command is:

```
discover network-id parameters
```

The **network-id** is the ID of the network object which will be the parent of the discovered devices.

The **parameters** specify the type of discovery, the IP addresses (or DNS names) to discover, and the security settings that will be required to authenticate to the device. The Type parameter defaults to Discover, so for an ordinary discovery this can be

omitted. The security settings are required in order to configure the device or to obtain the capabilities of specific interfaces, so if they are not set on discovery they must be set afterwards and the capabilities re-fetched.

Discovering a Single Device

To discover a single device, use the **discover** command specifying the IP address of the device to be discovered.

```
discover network-id ipaddress=10.0.0.22 mask=32
discover network-id DnsName=foo
```

This will discover the device, creating it as a child of the network object.

Discovering a Single Device Using an SNMP Profile

To discover a single device using an SNMP Profile, when the authentication option is selected:

```
discover network-id ipaddress=10.0.0.22 mask=32 SnmpProfileName=xxxxxxxx
```

Discovering a Subnet

To discover a whole subnet, use the **discover** command with the following format:

```
discover network-id ipaddress=subnet-address mask=24
```

Fetching Capabilities

To fetch device and interface capabilities, you need to provide security information for the devices. The exact command syntax depends on the type of security required.

Anonymous Telnet

```
discover network-id IpAddress=IPaddress AccessStyle=Anonymous
LoginPassword=loginpassword EnablePassword=enablepassword
```

Named User

```
discover network-id IpAddress=IPaddress AccessStyle=NamedUser UserName=username
LoginPassword=loginpassword EnablePassword=enablepassword
```

TACACS

To discover using TACACS, the UserName, ResponseStrings, and EnablePassword attributes of the Network object must be set first:

```
modify network-id UserName=username ResponseStrings=responsestrings
EnablePassword=enablepassword
commit
```

The device can then be discovered using a command as follows:

```
discover network-id IpAddress=IPaddress AccessStyle=TACACS
```

Rediscovering a Device

To rediscover an existing device, simply use the **discover** command with the ID of the device:

```
discover device-id
```

Stopping Discovery

To stop all discoveries in progress, use:

```
discover Type=Stop
```

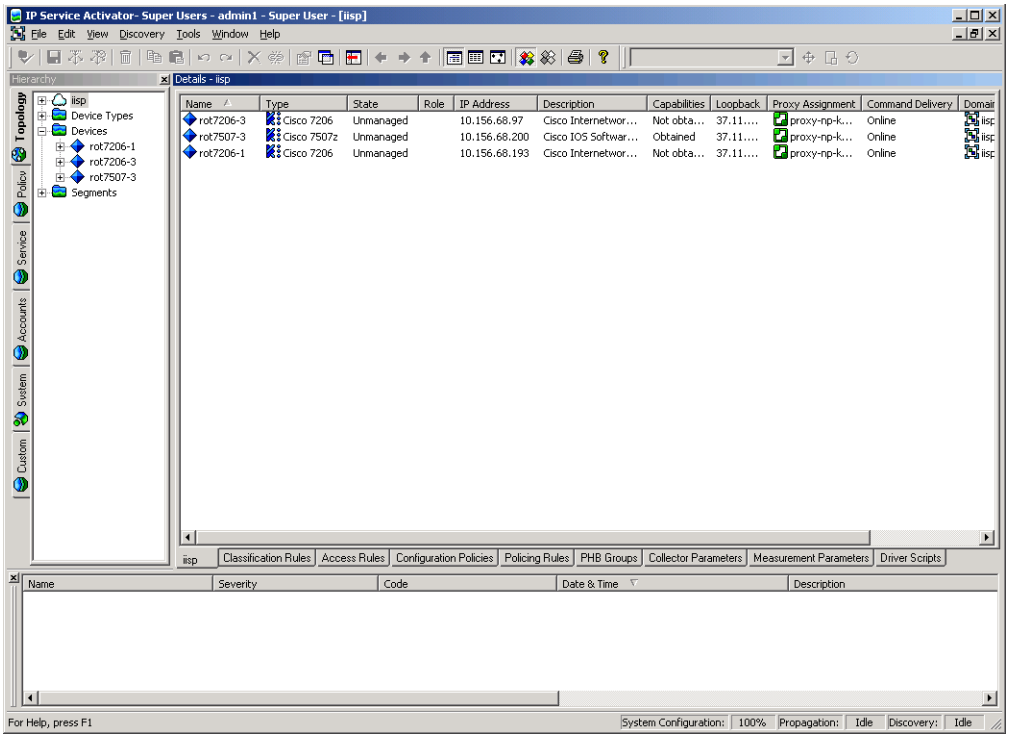
Example

In this example, three devices are discovered within a network (ID=18)

```
discover [18] IpAddress=10.0.0.91 AccessStyle=anonymous LoginPassword=cisco
EnablePassword=cisco
discover [18] IpAddress=10.0.0.92 AccessStyle=anonymous LoginPassword=cisco
EnablePassword=cisco
discover [18] IpAddress=10.0.0.93 AccessStyle=anonymous LoginPassword=cisco
EnablePassword=cisco
/>GetChildren [18]
[2707] Device:"rot7206-1"
[2731] Device:"rot7206-3"
[2799] Device:"rot7507-3"
```

The user interface is as in Figure 5–6.

Figure 5–6 Three Devices Within a Network



The three devices appear as children of the network object. As part of the discovery, any role assignment rules that have been set up using the user interface will also be applied to the newly-discovered objects.

Creating Device and Interface Objects for Pre-Provisioning

Device objects, and interfaces on those devices, can be created directly using the OIM to permit pre-provisioning.

Pre-requisites

- A domain must have been created using the user interface.

Command Syntax

If the required DeviceType object for the device does not exist, then it must be created as a child of the Topology object:

```
setPath /Topology:"Topology"
create DeviceType:"name"
modify DeviceType:"name" attributes
```

Devices and interfaces may be created in the same way as any other object using the **create** command. Note that Device and Interface objects contain a number of attributes whose values can only be set on creation. See "DeviceType Object", "Device Object", and "Interface Object".

```
setPath network-id
create Device:"device-name" IpAddr=IPaddress DeviceType=device-type-name
IsVirtual=False ...
modify Device:"device-name" Name=new-device-name
commit
```

device-type-name is the name of a DeviceType object that is a child of the Topology object.

Interface objects can then be created as children of the Device object:

```
setPath device-id
create Interface:"interface-name" IpAddr=IPaddress SubnetMask =subnet-mask ...
modify Interface:"interface-name" Speed=new-interface-speed
commit
```

Once the Device and Interface objects have been created, and the correct security attributes set, the real devices can be discovered and associated with the Device object by using the discover command:

```
discover device-id
```

Example

In this example a device and an interface are created. The device is created as a child of the network object (ID=18). This example assumes that the DeviceType object already exists as a child of the Topology object.

```
setPath [18]
create Device:"New Device" IpAddr=1.2.3.4 DeviceType="New Type" IsVirtual=False
commit
setPath Device:"New Device"
create Interface:"NewInterface" ipAddr=5.6.7.8 SubnetMask=255.255.255.0
commit
```

Creating and Assigning Roles to Devices and Interfaces

Before devices and interfaces can be configured, you need to assign roles to them in order to define the services and policies that can be applied to them. You can assign both system-defined roles and user-defined roles, and create new user-defined roles if required.

Pre-requisites

- The Device and Interface objects to which you are applying roles must already exist.

Command Syntax

This section describes the command syntax.

Creating a New Role

To create a new device or interface role, set the path to the Policy object, and use the **create** command to create the roles:

```
setPath /Policy:"Policy"
create RoleDevice:"new-device-role"
create RoleInterface:"new-interface-role"
```

Assigning a Role to a Device, Interface, or SubInterface

To assign a role (either a newly-created one, or an existing one) to a device, interface, or sub-interface simply link the role to the object.

```
link [object-id] [role-id]
```

Note that RoleDevice objects can only be children of Device objects, and RoleInterface objects can only be children of Interface or SubInterface objects.

Example

In this example, roles are assigned to the three devices discovered in "Discovering Devices". The PE device is assigned a role of “Gateway” while the two CE devices are assigned the role of “Access”. The interfaces linking the devices are all assigned the “Access” interface role.

Object IDs are as follows:

```
RoleDevice:"Gateway"=[5]
RoleDevice:"Access"=[4]
RoleInterface:"Access"=[11]
```

To assign appropriate roles to the devices:

```
link [2799] /Policy:"Policy"/RoleDevice:"Gateway"
link [2707] /Policy:"Policy"/RoleDevice:"Access"
link [2731] /Policy:"Policy"/RoleDevice:"Access"
commit
```

Alternatively, use the object IDs, such as:

```
link [2731] [4]
```

To assign appropriate roles to the interfaces:

```
link [2865] [11]
link [2867] [11]
link [2838] [11]
link [2962] [11]
commit
```

Creating a Site

This section explains the steps involved in creating a site and setting up the routing parameters.

Pre-requisites

- Sites are associated with PE Access interfaces. They can also be associated with CE devices if they are to be managed, and PE devices, when creating an interface-less site. These devices and interfaces must have been discovered and appropriate roles must be assigned.

Command Syntax

This section describes the command syntax.

Create Customer Object

Sites must be created as children of Customer objects, so you need to create a customer object as a child of the Domain if it does not already exist.

```
setPath domain-id
create Customer:"name"
commit
```

Create Site Object

The site can then be created as a child of the required customer:

```
setPath customer-id
create Site:"name"
commit
```

Link PE Access Interface to Site

The appropriate interface on the PE device needs to be linked to the site:

```
link site-id interface-id
commit
```

Link CE Router to Management Site

If the site is to contain a CE router, then that must also be linked to the site:

```
link site-id ce-device-id
commit
```

Manage the CE Router

If the site contains the CE router, then it should be managed.

```
manage ce-device-id
commit
```

Setting Up PE-CE Routing Parameters

The routing parameters between the PE and the CE can then be set. This involves modifying certain attributes on the site.

Routing Protocol

To configure the appropriate protocol, the Routing Protocol must be set to **RIP**, **EBGP**, **OSPF**, or **NONE**.

```
modify site-id RoutingProtocol=value
commit
```

For RIP and OSPF, no further parameters need to be set. If the RoutingProtocol is set to **EBGP**, then additional attributes should be set.

- The BgpAsn attribute on the Site object should be set to the ASN of this site.
- The PrivateCeIpAddress attribute on the Interface object should be set to the IP address of the access interface on the CE.

```
modify management-site-id BgpAsn=value
modify interface-id PrivateCeIpAddress=value
commit
```

Static Routing

Setting the InstallStatic attribute on the Site object to True will specify static routing. The destinations that can be reached from the PE Access interface can then be set by creating StaticRoute objects as children of the PE Access interface and setting their attributes:

```
setPath interface-id
create StaticRoute:""
modify static-route-id ipaddr=ce-ipaddr subnetmask=mask
nexthop=ce-access-interface-ipaddr
commit
```

If multiple static routes are available then a StaticRoute object should be created for each destination accessible from the PE Access interface.

Example

In this example a site is created that contains the PE access interface "Ethernet 0/1" and the CE device "A1720-1". This site is used as the management site of a management VPN.

The routing protocol for the site is EBGP, and the address of the access interface on the CE device is 10.136.135.2. Static routing to the CE device is also configured.

OIM Commands

First, the customer object is created as a child of the Domain object (ID=30):

```
setPath [30]
create Customer:"Management"
commit
```

Next the Site object is created as a child of the newly-created customer:

```
setPath Customer:"Management"
create Site:"Management Hub"
commit
```

Now the PE access interface (ID=2867) and the CE device (ID=2837) must be linked to the site:

```
link Site:"Management Hub" [2867]
link Site:"Management Hub" [2837]
commit
```

Now the CE and PE devices can be managed:

```
manage [2837]
manage [2861]
commit
```

Next the routing protocol for the site should be set to EBGP, and the BgpAsn attribute is set to a nominal value (in this case 60).

```
modify Site:"Management Hub" RoutingProtocol=EBGP BgpAsn=60
commit
```

The PrivateCeIpAddr attribute on the PE access interface (ID=2867) in the site should be set to the IP address of the CE's access interface.

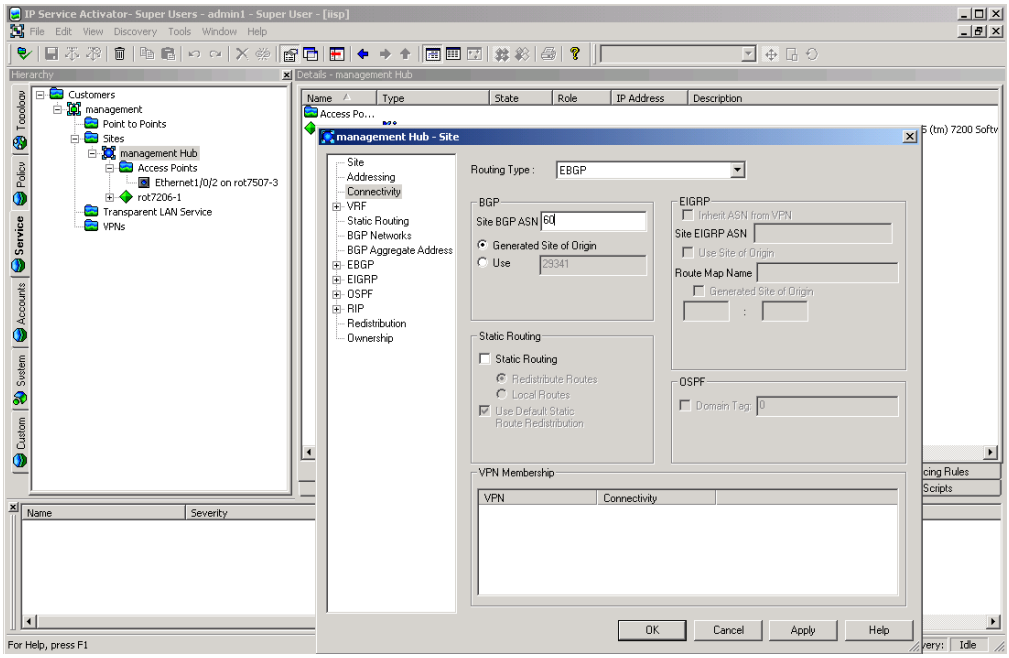
```
modify [2867] PrivateCeIpAddr=10.135.135.2
commit
```

Now the static route is configured. A StaticRoute object is created as a child of the PE access interface (ID=2867), and the attributes of the object are set accordingly.

```
setPath [2867]
create StaticRoute:"0.0.0.0"
modify StaticRoute:"0.0.0.0" ipaddr=10.0.0.91 subnetmask=255.255.255.255
nexthop=10.135.136.2
commit
```

Figure 5–7 shows that a customer and a site have been created. The **Connectivity** properties page shows that the ASN has been correctly set.

Figure 5–7 The Connectivity Properties Page



Creating a Management VPN

This section explains the steps involved in setting up a management VPN.

Pre-requisites

- The management VPN must contain a site that will be the hub of the VPN, and any customer sites that are also present in their own VPNs. These sites must have been previously set up. It is usual to create a management customer that will contain the management site and the management VPN object.

Command Syntax

This section describes the command syntax.

Creating the VPN Object

The first step is to create a VPN object as a child of the management customer:

```
setPath customer-id
create VPN:"ManagementVPN"
commit
```

The VPN must be configured as a management VPN:

```
modify vpn-id MplsVpnType=Management
commit
```

Link the Management Site to the VPN, and Make it a Hub Site

The Management Site object can now be linked to the VPN. This site should be set up as a hub site of the management VPN. To do this, after linking the site to the VPN, create a SiteHub object as a child of the VPN with the same name as the management site:

```
link vpn-id management-site-id
setPath vpn-id
create SiteHub:"management-site-name"
commit
```

Link Customer Sites to the VPN

So far only one site has been added to the management VPN. Any other sites linked to customer VPNs should also be linked to the management VPN object:

```
link vpn-id customer-site-id
commit
```

Example

In this example, a management VPN is created for the Management customer. The Management site is added to the VPN, and configured to be a hub. A single customer site is then added to the management VPN as a spoke.

Firstly the management VPN object should be created as a child of the Management customer (ID=3000).

```
setPath [3000]
create VPN:"Management VPN"
modify VPN:"Management VPN" MplsVpntype=Management
commit
```

Now the management site (ID=3002) should be linked to the VPN object:

```
link VPN:"Management VPN" [3002]
commit
```

The management site is made a hub of the VPN, by creating a SiteHub object with the name “Management Site” as a child of the VPN.

```
setPath VPN:"Management VPN"
create SiteHub:"Management Hub"
commit
```

Finally any customer sites should be linked to the VPN object. In this example a single customer site object (ID=3013) is linked to the VPN object (ID=3020).

```
link [3020] [3013]
commit
```

Figure 5–8 shows the management VPN containing the two sites.

Figure 5–8 Management VPN

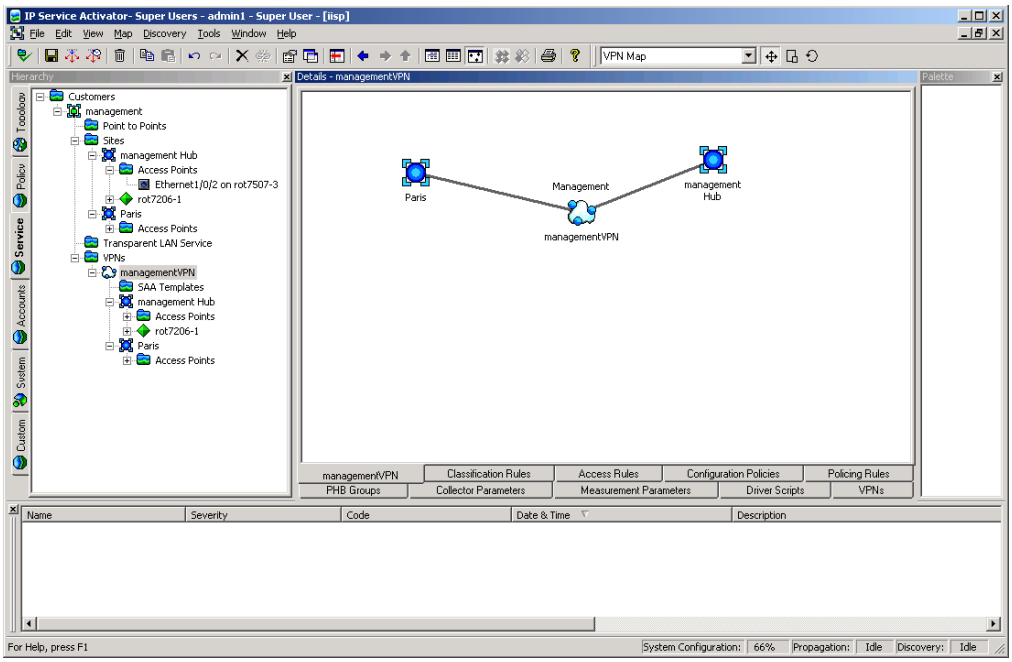
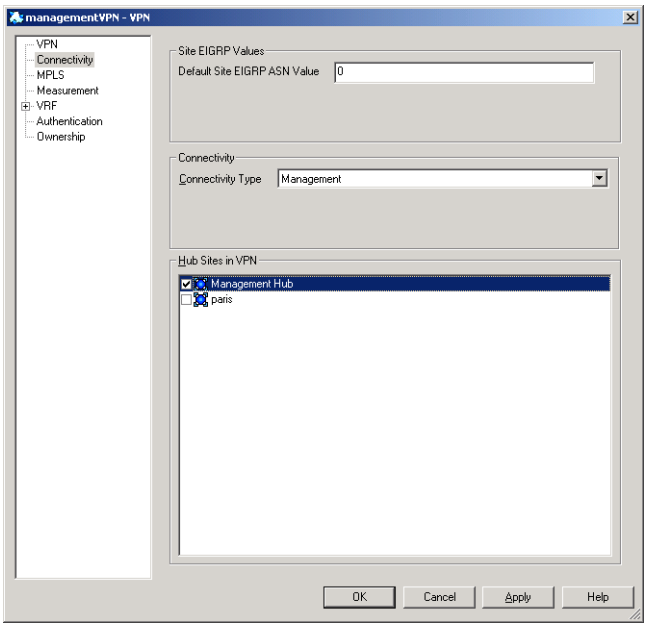


Figure 5–9 shows the properties box of the VPN, showing that “Management Site” is a hub, and “Paris” is a spoke.

Figure 5–9 Management VPN Properties Box



Creating a Customer VPN

This section explains the steps involved in creating a customer VPN.

Pre-requisites

- Sites to be included in the VPN must be created and configured as described above.

Command Syntax

This section describes the command syntax.

Create the VPN Object

A VPN object must be created as a child of the customer:

```
setPath customer-id
create VPN:"vpn-name"
commit
```

The VPN object must be configured as a either a hub and spoke or a full mesh VPN:

```
modify [vpn-id] MplsVpnType=type
commit
```

Link the Customer Site to the VPN

Any sites that are to be in the VPN must be linked to the VPN object:

```
link customer-vpn-id customer-site-id
commit
```

Repeat this step for all sites to be included in the VPN.

Set Hub Sites

If the customer VPN is hub and spoke, one of the linked sites must be defined as the hub site. This is achieved by creating a SiteHub object as a child of the VPN object, with the same name as the site. The *hub-site-name* must match the name of the site under the hub and spoke VPN.

```
setPath customer-vpn-id
create SiteHub:"hub-site-name"
commit
```

If the VPN is full mesh VPN, this step is not necessary.

To remove a SiteHub which is linked to a hub in a hub and spoke VPN:

```
unlink [vpn id] [sitehub id]
delete [sitehub id]
commit together
```

Link Customer Sites to the Management VPN

Any sites linked to any customer VPN must also be linked to the management VPN object, if this has not already been done:

```
link management-vpn-id customer-site-id
commit
```

This should be repeated so all the customer sites are linked to the management VPN object.

Example

This example creates a full mesh VPN containing three customer sites (assumed to be already created, but not yet linked to the management VPN).

OIM Commands

First a VPN object is created as a child of Customer 1 (ID=3012).

```
setPath [3012]
create VPN:"Acme Customer VPN"
commit
```

The VPN is defined as full mesh:

```
modify VPN:"Acme Customer VPN" MplsVpnType=FullMesh
commit
```

The three customer sites are linked to the VPN object:

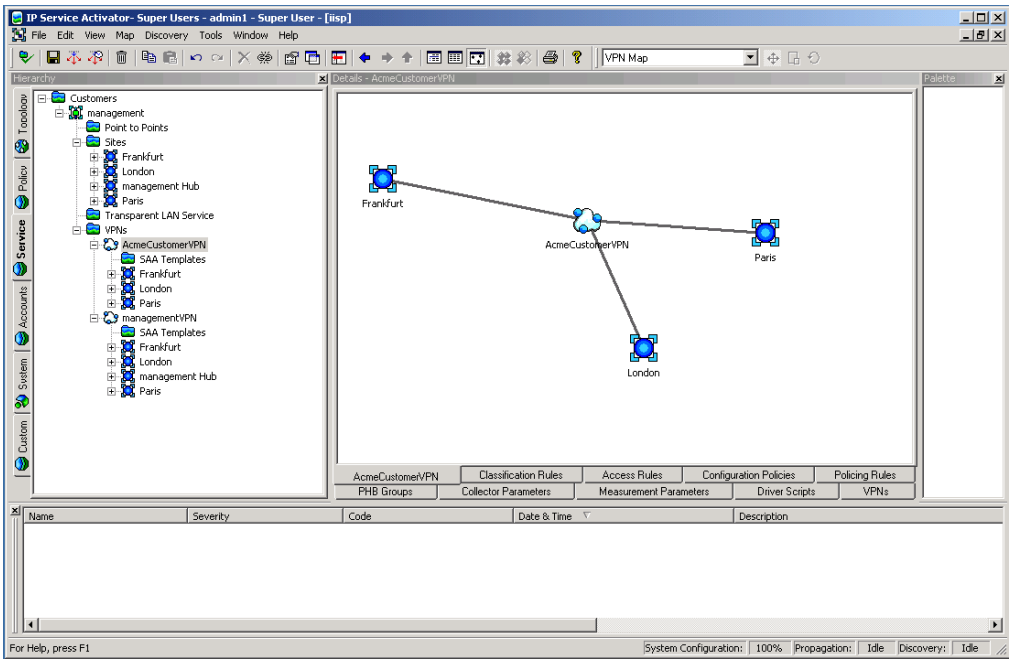
```
link VPN:"Acme Customer VPN" Site:"London"
link VPN:"Acme Customer VPN" Site:"Paris"
link VPN:"Acme Customer VPN" Site:"Frankfurt"
commit
```

The three customer sites are also linked to the management VPN (ID=3084).

```
link [3084] Site:"London"
link [3084] Site:"Paris"
link [3084] Site:"Frankfurt"
commit
```

Figure 5–10 shows the Customer VPN containing three sites.

Figure 5–10 Customer VPN



Creating and Applying Rules

This section describes how to create and configure classification rules, access rules and policing rules.

Creating a rule involves the following steps:

- Create the RuleClassification, RuleAccess, or RulePolicing object.
- Link RoleDevice and RoleInterface objects to the Rule to set the role(s) that the rule will apply to.
- Set the attributes of the Rule object.

Once all these steps have been performed, and the final commit executed, then ConcreteRule objects appear as children of any interfaces that match the role criteria.

Pre-requisites

- The devices and interfaces on which the rules are applied must have already been discovered, and must have roles assigned to them.

Command Syntax

This section describes the command syntax.

Creating the Rule

Consider where the rule is to be applied. If it is to apply to all devices and interfaces that match the role(s), the rule should be created as a child of the Domain object. If the rule is only to apply on the interfaces of one specific device, then the rule should be created as a child of that device:

```
setPath domain-id
create RuleType:"rule-name"
```

```
commit
setPath device-id
create RuleType:"rule-name"
commit
```

RuleType may be RuleClassification, RuleAccess, or RulePolicing.

> **Note:** The rule applies to global domain only when that domain contains VPN and its applying to only that device which is part of the **VPN**.

Assigning Roles to the Rule

To assign roles that the rule will apply to, simply link the appropriate RoleDevice and RoleInterface objects to the rule:

```
link rule-id role-id
commit
```

Rule Attributes: Inbound/Outbound

To specify whether the rule applies inbound, outbound or both, set the InBound and OutBound flags to true or false using the **modify** command:

```
modify rule-id InBound=true
modify rule-id OutBound=true
commit
```

Rule Attributes: Date and Time Period

You can set an applicable date and time period in one of two ways.

One method is to modify the StartTime, EndTime, and DaysOfWeek attributes using the **modify** command. StartTime and EndTime must be set to a string with the format "YYYY/MM/DD HH:MM:SS". The DaysOfWeek attribute uses an unsigned integer where each bit represents a day of the week. For example:

```
modify rule-id StartTime="2001/10/10 10:00:00"
modify rule-id EndTime="2001/10/10 10:00:00"
modify rule-id DaysOfWeek="127"
commit
```

The alternative is to modify the PeriodName attribute to match the name of an existing Period object (which must be a child of the Domain object). When this method of setting the data and time is used, any change to the Period object changes the start and end time of the Rule object.

```
modify rule-id PeriodName="period-object-name"
commit
```

Rule Attributes: Classification

The Classification of the rule can be set in one of two ways, using local classification or global classification.

To set up local classification, modify the Classification object (which must be a child of the Rule object) to ensure the appropriate source/destination IP address and account name are set. Also modify the TrafficName attribute to the name of an existing Traffic object if required. For example, to set the Rule to work on traffic from IP address

10.0.0.3, to a destination account *dest-account* and traffic type *traffic-name* use the following commands:

```
modify classification-id SourceIpAddr=10.0.0.3
modify classification-id SourceMask=255.255.255.255
modify classification-id DstAccountName="dest-account"
modify classification-id TrafficName="traffic-name"
commit
```

To set up global classification, first delete the Classification object that is a child of the Rule. Then create a ClassificationGroup object as a child of the Rule. Finally, any number of existing Classification objects or ClassificationGroup objects may be linked to the newly created ClassificationGroup object. For example:

```
setPath rule-id
delete classification-id
create ClassificationGroup:"name"
link classification-group-id classification-object
commit
```

The remaining attributes differ depending on the type of rule.

Classification Rule

To set packet marking on a classification rule, modify the PacketMarking attribute to the name of a PacketMarking object that is a child of the Domain object:

```
modify rule-id PacketMarkingName="packet-marking-name"
commit
```

Creating and Linking a Packet Marking to a Traffic Type

To create a packet marking and link it to a traffic type:

```
create TrafficPacketMarking:"TrafficTypeTT1"
create PacketMarking:"TrafficPM1"
link [traffic-type-id] [packet-marking-id]
commit
```

Note: the traffic type specified must match the name of the existing traffic type.

To unlink a packet marking from a traffic type:

```
modify [classification-id] TrafficName=""
commit
```

Access Rule

Set the Permit attribute to True to permit the identified traffic, or to False to deny the identified traffic:

```
modify rule-id Permit=false
commit
```

Policing Rule

Set the CommittedRate, NormalBurstSize, and ExcessBurstSize attributes using the **modify** command:

```
modify rule-id CommittedRate=8000
modify rule-id NormalBurstSize=1000
modify rule-id ExcessBurstSize=2000
commit
```

Example

This example creates rules to apply QoS on “Acme Customer VPN” which was set up in an earlier example.

A classification rule is created that marks real-time traffic within Acme Customer VPN is marked with a specific IP precedence value, and an access rule is created to prevent any Game traffic (Quake) from traveling on the VPN.

Classification Rule

A RuleClassification object is created as a child of the VPN (ID= 3105):

```
setPath [3105]
create RuleClassification:"Real-time"
commit
```

The device and interface roles are set up to indicate which interfaces the rule will apply to. In this case the rule is applied to the Access interfaces on the PE. Therefore the RuleClassification object is linked to the Gateway device role (ID=5) and the Access interface role (ID=11):

```
link RuleClassification:"Real-time" [5]
link RuleClassification:"Real-time" [11]
commit
```

The rule is to apply to inbound traffic on the interfaces, so the Rule’s InBound attribute is set to True, and the OutBound attribute is set to False.

```
modify RuleClassification:"Real-time" InBound=True OutBound=False
commit
```

Next the packet marking is set. The PacketMarkingName attribute is set to "IP Prec 5", which is the name of the PacketMarking object that has been assigned to the identified traffic.

```
modify RuleClassification:"Real-time" PacketMarkingName="IP Prec 5"
commit
```

Concrete rule objects will now appear as children of the three PE access interfaces in the VPN.

Access Rule

An access rule is set up to prevent Quake traffic from using the VPN. This rule will be configured to apply to the PE Access interfaces in the VPN.

A RuleAccess object is created as a child of the customer VPN (ID=3105):

```
setPath [3105]
create RuleAccess:"Quake Deny"
commit
```

Now the RuleAccess object is linked to the Gateway device role (ID=5) and the Access interface role (ID=11).

```
link RuleAccess:"Quake Deny" [5]
link RuleAccess:"Quake Deny" [11]
commit
```

The rule is to apply to inbound traffic on the interfaces, so the rule’s InBound attribute is set to True, and the OutBound attribute is set to False.

```
modify RuleAccess:"Quake Deny" InBound=True OutBound=False
commit
```

As the access rule will deny traffic, the Permit attribute should be set to False:

```
modify RuleAccess:"Quake Deny" Permit=False
commit
```

Finally, the traffic type which identifies the traffic to be denied is set. To do this the TrafficName attribute of the Classification object that is a child of the RuleAccess object is set to the name of the traffic to be denied:

```
modify RuleAccess:"Quake Deny"/Classification:"" TrafficName="quake-c"
commit
```

Creating and Applying PHB Groups

Creating and apply PHB groups involves the following steps:

- Create a PHBGroup object
- Modify the PhbGroup object's PHB child object

  The OIM automatically creates a PHB child object when you create a PHBGroup object.
- Create a PHB object for each class of service to be handled by the PHBGroup object

Pre-requisites

- Any Cos object to be used by an MQC PHB group must be linked to a Classification object

Creating a Standard PHB Group

A PHB group must always be created either as a child of the Domain object or as a child of a PHBGroupFolder, which itself is a child of the Domain object. When a PHBGroup object is created, a PHB child object is automatically created and its ClassName attribute set to 'Default Class of Service'. The Name attribute is automatically taken from the ClassName attribute; the two are inextricably linked.

```
setpath /Policy:"Policy"/Domain:"USA"
create PHBGroup:"WRRPHB"
modify PHBGroup:"WRRPHB" Action=8
commit
```

The device and interface roles are set up to indicate which interfaces the PHB group will apply to. In this case the PHB group is applied to the Access interfaces on the PE. Therefore the PHBGroup object is linked to the Gateway device role (ID=5) and the Access interface role (ID=11):

```
link PHBGroup:"WRRPHB" [5]
link PHBGroup:"WRRPHB" [11]
commit
```

PHBAtm and PHBFrts objects are also automatically created as children of the PHBGroup object:

```
setpath PHBGroup:"WRRPHB"
getchildren
```

```
    [1197] PhbAtm:"PhbAtm"
    [1196] PhbFrts:"PhbFrts"
    [1190] Phb:"Default Class of Service"
```

Deleting the Default Class of Service and Linking to an Existing Class of Service

When a PHBGroup object is created, a PHB child object is automatically created and its ClassName attribute set to 'Default Class of Service'. You can delete this object and link to an existing CoS.

To delete the default CoS and link to an existing CoS:

1. Use the following command syntax:

```
delete [default_cos_id]
setpath [phb_id]
create Phb:"Gold" classname="Gold"
commit
```

   where *phb_id* is the ID of the PHB Group to be linked to a new CoS, *default_cos_id* is the ID of the default CoS, and Gold is the name of the existing CoS to be linked.

The PHBGroup automatically inherits the name of the new CoS.

If the new CoS needs a different CoS mechanism, set it before committing.

> **Note:** If you simply unlink the default CoS rather than deleting it, the default will become an orphan, which generates an error.

Creating Additional PHB Children

You can create additional PHB objects as children of the PHBGroup object. You must set the ClassName attribute at creation:

```
setpath /Policy:"Policy"/Domain:"USA"/PHBGroup:"WRRPHB"
create PHB:"name" ClassName="Silver"
commit
modify PHB:"Silver" ClassName="Gold"
commit
```

Note that the value assigned to the PHB object's **name** attribute is ignored by the OIM. The OIM automatically assigns a name to the PHB object, using the value specified in the ClassName attribute; that is, the PHB object's name is taken from the CoS to which it is linked.

If no ClassName is specified at creation, the attribute is automatically assigned the string 'Default Class of Service'. Assigning the same ClassName value to two PHB objects results in an error:

```
CommandExecutionError.ERR_ObjectModelViolation: Object Cos is linked to more than
one Phb under the same PHBGroup [PhbGroupId].
```

Use the **getchildren** command to view the children of the PHBGroup object 'WRRPHB':

```
/Policy:"Policy"/Domain:"USA"/PHBGroup:"WRRPHB">getchildren
    [1197] PhbAtm:"PhbAtm"
    [1196] PhbFrts:"PhbFrts"
    [1190] Phb:"Default Class of Service"
    [1205] Phb:"Gold"
```

Creating an MQC PHB Group

An MQC PHB group must be created either as a child of the Domain object or as a child of a PHBGroupFolder, which itself is a child of the Domain object, in the same way as a standard PHB group.

When a PHBGroupMqc object is created with the attribute CreateDefaultPhbMqc set to its default value of True, the OIM automatically creates a PhbMqc child object. The child object's ClassName attribute is set to 'Default Class of Service', which associates the object with this CoS. It is essential to modify the CosName attribute and specify the QoS action:

```
setpath /Policy:"Policy"/Domain:"USA"
create PHBGroupMqc:"CBWFQMQC"
modify PHBGroupMqc:"CBWFQMQC"/PhbMqc:"Default Class of Service" …
CosName="Gold" Action=2
commit
```

Refer to Table 4–73, " PHBMqc Object Attributes" for a list of applicable Action types.

Note that if a value is not specified for the Action attribute the following error is generated:

```
CommandExecutionError.ERR_ObjectModelViolation: [3653], No MQC Action has been
selected.
```

The device and interface roles are set up to indicate which interfaces the MQC PHB group will apply to. In this case the MQC PHB group is applied to the Access interfaces on the PE. Therefore the PHBGroupMqc object is linked to the Gateway device role (ID=5) and the Access interface role (ID=11):

```
link PHBGroupMqc:"CBWFQMQC" [5]
link PHBGroupMqc:"CBWFQMQC" [11]
commit
```

Deleting the Default Class of Service and Linking to an Existing Class of Service

When a PHBGroupMqc object is created, a PHBMqc child object is automatically created and its ClassName attribute set to 'Default Class of Service'. You can delete this object and link to an existing CoS.

To delete the default CoS and link to an existing CoS:

1. Use the following command syntax:

   ```
   delete [default_cos_id]
   setpath [phbmqc_id]
   create Phb:"Gold" classname="Gold"
   commit
   ```

   where *phbmqc_id* is the ID of the PHBGroupMqc to be linked to a new CoS, *default_cos_id* is the ID of the default CoS, and Gold is the name of the existing CoS to be linked.

The PHBGroupMqc automatically inherits the name of the new CoS.

If the new CoS needs a different CoS mechanism, set it before committing.

> **Note:** If you simply unlink the default CoS rather than deleting it, the default will become an orphan, which generates an error.

Creating Additional PHBMqc Children

You can create additional PHBMqc objects as children of the PHBGroupMqc object. You must set the CosName attribute at creation:

```
setpath /Policy:"Policy"/Domain:"Test"/PhbGroupMqc:"CBWFQMQC">
create PHBMqc:"PHBMQCSilver" CosName="Silver" Action=16
commit
```

If no CosName is specified at creation, the attribute is automatically assigned the string 'Default Class of Service'. Assigning the same CosName value to two PHBMqc objects results in an error.

Implementing Single or Two-rate Policing with a PHBMqc Object

Implementing single or two-rate policing with a PHBMqc object involves the following attributes:

- Action: set to 4 for single-rate policing, 8 for two-rate policing
- ConformAction: specify the name of a PHBPolicingAction object
- ExceedAction: specify the name of a PHBPolicingAction object
- ViolateAction: specify the name of a PHBPolicingAction object

For example, to implement two-rate policing:

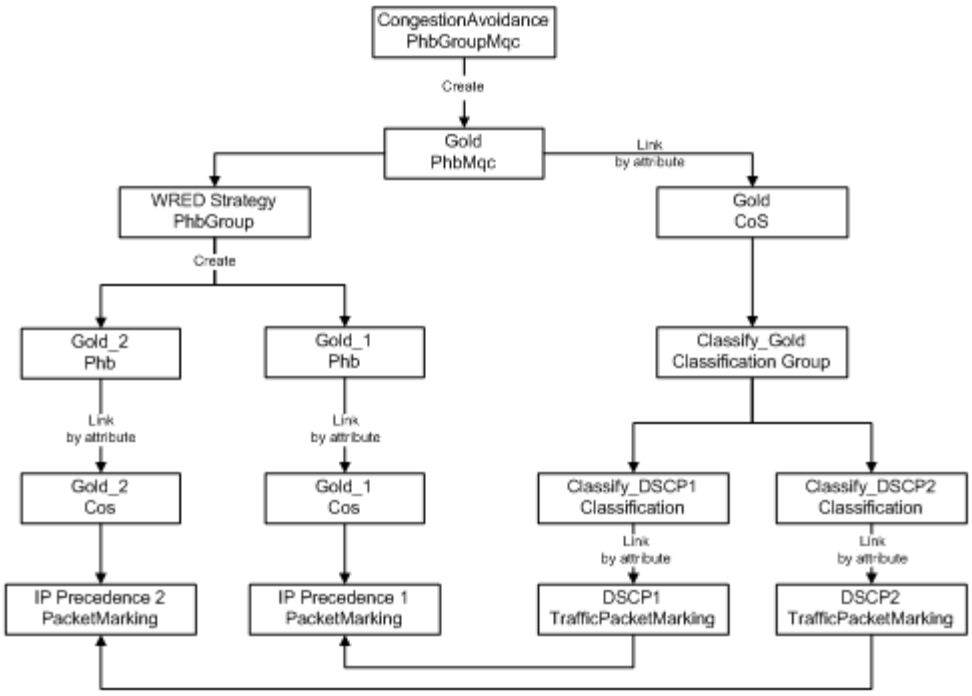
```
modify PhbMqc:"Default Class of Service" Action=8
ConformAction="Default Conform Action"
ExceedAction="Default Exceed Action"
ViolateAction="Default Violate Action"
```

For a list of applicable Action values, refer to "PHBGroup Objects".

Configuring WFQ Using WRED as the Drop Strategy

It is possible to configure WFQ using WRED as the drop strategy. Figure 5–11 illustrates the object relationships that must be created to configure WFQ+WRED.

Figure 5–11 Object Relationships for WFQ+WRED Configuration



Pre-requisites

The example assumes the following:

- PacketMarking objects IP Precedence 1 and IP Precedence 2 have been created through the user interface

  It is also possible to perform this step via the OIM. However, these packet markings are created automatically by loading the **default.dscp.policy** file into IP Service Activator.

Sample Code

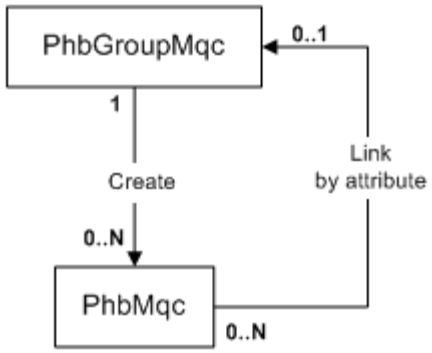
```
*** The PhbMqc requires that any Cos it applies to is linked to a
*** Classification so we must create an appropriate Classification. In
*** this case it will be a ClassificationGroup containing
*** TrafficPacketMarkings DSCP1 and DSCP2.
create TrafficPacketMarking:"DSCP1"
create TrafficPacketMarking:"DSCP2"
link TrafficPacketMarking:"DSCP1" PacketMarking:"IP Precedence 1"
link TrafficPacketMarking:"DSCP2" PacketMarking:"IP Precedence 2"

create Classification:"Classify_DSCP1" TrafficName="DSCP1"
create Classification:"Classify_DSCP2" TrafficName="DSCP2"

create ClassificationGroup:"Classify_Gold"

link ClassificationGroup:"Classify_Gold" Classification:"Classify_DSCP1"
link ClassificationGroup:"Classify_Gold" Classification:"Classify_DSCP2"

create Cos:"Gold"
link Cos:"Gold" ClassificationGroup:"Classify_Gold"
```

```
*** Create the 2 Cos used in the WRED Phbs that will determine the WRED
*** strategy.

create Cos:"Gold_1"
create Cos:"Gold_2"

link Cos:"Gold_1" PacketMarking:"IP Precedence 1"
link Cos:"Gold_2" PacketMarking:"IP Precedence 2"

*** Create the WRED PhbGroup

create PhbGroup:"WRED Strategy" Action=8

create PhbGroup:"WRED Strategy"/Phb:"Gold_1" ClassName="Gold_1"
create PhbGroup:"WRED Strategy"/Phb:"Gold_2" ClassName="Gold_2"

*** Commit the transaction
commit

*** These are the children of the WRED PhbGroup.
*** One can modify the attributes for the Phbs as required (or could set
*** them during creation).

/Policy:"Policy"/Domain:"Domain1"/PHBGroup:"WRED Strategy">getchildren
    [1583] PhbAtm:"PhbAtm"
    [1582] PhbFrts:"PhbFrts"
    [1576] Phb:"Default Class of Service"
    [1590] Phb:"Gold_2"
    [1584] Phb:"Gold_1"

*** Create the PhbGroupMqc (Action = 66 = 2 + 64 = CBWFQ + Congestion)

setpath /Policy:"Policy"/Domain:"Domain1"
create PhbGroupMqc:"Congestion Avoidance"
modify PhbGroupMqc:"Congestion Avoidance"/PhbMqc:"Default Class of Service"
CosName="Gold" Action=66
modify PhbGroupMqc:"Congestion Avoidance"/PhbMqc:"Gold" WredStrategy=WRED
link PhbGroupMqc:"Congestion Avoidance"/PhbMqc:"Gold" PHBGroup:"WRED Strategy"

commit
```

Nesting MQC PHB Groups

A PHBGroupMqc object that specifies CBWFQ, Shaping or Policing may nest a child PHBGroupMqc and so configure what Cisco refers to as a 'hierarchical service policy'. IP Service Activator now allows the user to select the Marking Action along with Nesting Action, in addition to the already existing LLQ, CBWFQ, Shaping and Policing actions. Figure 5–12 illustrates the object relationships that must be formed to nest MQC PHB groups.

Figure 5–12 Object Relationship for Nesting MQC PHB Groups



Pre-requisites

The example assumes the following:

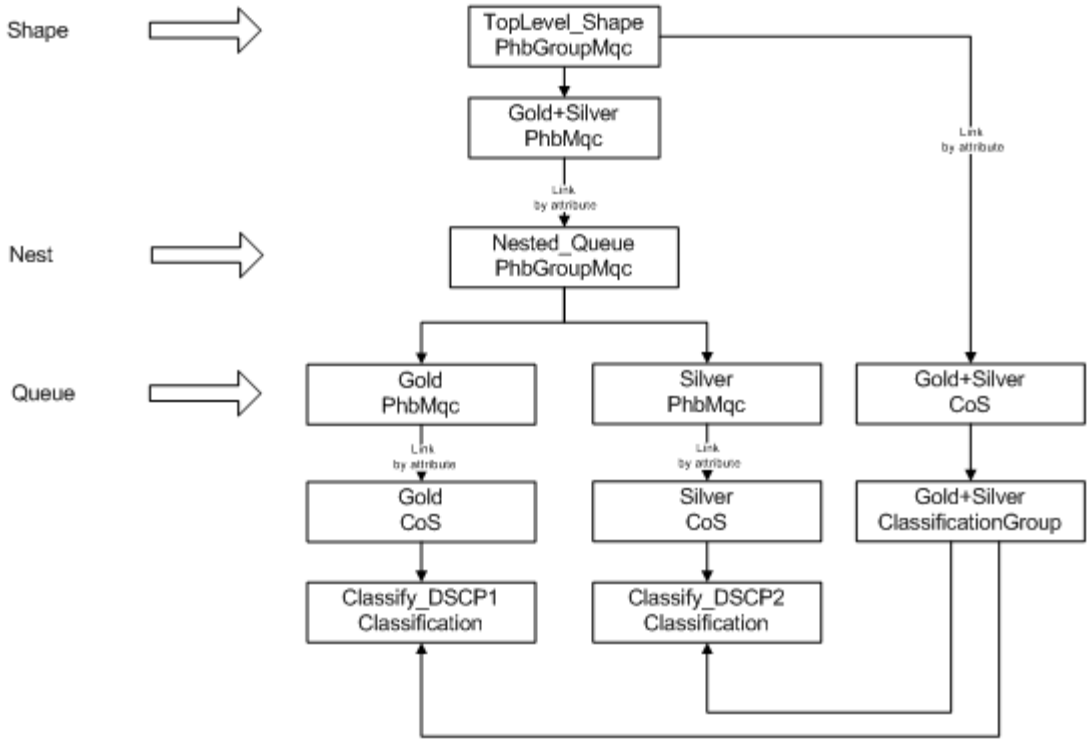
- PacketMarking objects IP Precedence 1 and IP Precedence 2 have been created through the user interface

  It is also possible to perform this step via the OIM. However, these packet markings are created automatically by loading the **default.dscp.policy** file into IP Service Activator.

Overview of the Example

This example uses nested MQC PHB groups to apply traffic shaping to incoming Gold and Silver traffic and different queuing treatments to each class of service individually. The object relationships that are created and the policies that are applied are shown in Figure 5–13.

Figure 5–13 Object Relationships and Policies



Sample Code

```
*** The PhbMqc requires that the Cos is linked to a Classification so we
*** must create an appropriate classification. To apply shaping to
*** Gold and Silver traffic we need to create a ClassificationGroup
*** containing 2 types of traffic - DSCP1 and DSCP2.

create TrafficPacketMarking:"DSCP1"
create TrafficPacketMarking:"DSCP2"
link TrafficPacketMarking:"DSCP1" PacketMarking:"IP Precedence 1"
link TrafficPacketMarking:"DSCP2" PacketMarking:"IP Precedence 2"

create Classification:"Classify_DSCP1" TrafficName="DSCP1"
create Classification:"Classify_DSCP2" TrafficName="DSCP2"

create ClassificationGroup:"Gold+Silver"

link ClassificationGroup:"Gold+Silver" Classification:"Classify_DSCP1"
link ClassificationGroup:"Gold+Silver" Classification:"Classify_DSCP2"

create Cos:"Gold+Silver"
link Cos:"Gold+Silver" ClassificationGroup:"Gold+Silver"


*** We also need to create a Cos for each traffic type we
*** want to apply individual queuing parameters to.
create Cos:"Gold"
create Cos:"Silver"
link Cos:"Gold" Classification:"Classify_DSCP1"
link Cos:"Silver" Classification:"Classify_DSCP2"

*** Create a PhbGroupMqc specifying CB-WFQ as the treatment. This
```

```
*** PhbGroupMqc will be nested in a parent PhbGroupMqc that defines
*** traffic shaping parameters for the Cos's.

create PhbGroupMqc:"Nested_Queue"
modify PhbGroupMqc:"Nested_Queue" WfqBandwidthType=Percentage

*** We need to create 2 PhbMqcs that define queuing parameters for
*** each Cos - PhbMqc:"Gold" (renames the automatically created
*** "Default Class of Service" PhbMqc) and PhbMqc:"Silver"
*** Action=2 (CBWFQ) and we allocate different bandwidth for each
*** stream (35% and 20%)

modify PhbGroupMqc:"Nested_Queue"/PhbMqc:"Default Class of Service" CosName="Gold"
Action=2 WFQWeight=35
create PhbGroupMqc:"Nested_Queue"/PhbMqc:"Silver" CosName="Silver" Action=2
WFQWeight=20

*** Create the top-level PhbGroupMqc to shape both Cos's
*** (Gold+Silver) and select the nested PhbGroupMqc (linked by
*** attribute). Action= 144 (Shape + Nest)
create PhbGroupMqc:"TopLevel_Shape"
modify PhbGroupMqc:"TopLevel_Shape"/PhbMqc:"Default Class of Service"
CosName="Gold+Silver" Action=144 NestedPhbGroup="Nested_Queue"

commit
```

Creating an Event Subscription

This section describes how to create an event subscription that will collect events of a specified type occurring on specified objects.

Pre-requisites

- The objects on which events are to be collected must already exist and have been configured.

Command Syntax

Creating a subscription requires two steps:

1. An EventSubscription object must be created as a child of the System object. The EventSubscription specifies the delivery target.
2. One or more EventCollector objects must be created as a child of the EventSubscription. The EventCollector objects determine which type of events should be captured, and on which object(s).

Creating the Event Subscription

First the EventSubscription object is created as a child of the System object:

```
setPath /System:"System"
create EventSubscription:"name"
commit
```

Then the attributes of the EventSubscription must be set, namely the DeliveryType and the DeliveryDetails. The DeliveryType attribute indicates what sort of trap (if any) should be sent, and the DeliveryDetails attribute tells where to deliver the trap.

```
modify event-subscription-id DeliveryType=delivery-type
```

```
DeliveryDetails=delivery-details
commit
```

delivery-type can take the values SnmpTrap, UpgradedSnmpTrap, NetCool, CORBA Channel, or DatabaseOnly.

delivery-details is a string containing delivery details. For SNMP traps this is a comma-separated list specifying the IP address, port to which the traps are sent, and SNMP version (for example, "192.168.1.2,162,2"). For CORBA or NetCool, the string indicates the name of the service. If DeliveryType is DatabaseOnly the string should be blank.

Creating the Event Collector

Multiple EventCollector objects may be created as children of the EventSubscription object. Each EventCollector object traps one particular type of event on a group of objects. Therefore if, for example, you need to trap attribute changes and fault events occurring on a single object, two EventCollector objects would be required.

```
setpath /System:"System"/EventSubscription:"name"
create EventCollector:"name"
commit
```

After creation, set the type of event to be reported:

```
modify event-subscription-id Type=type
commit
```

type takes the values CreateAndDelete, Fault, AttributeChange, or StateChange.

Then set the object(s) to trap on. There are several ways to do this.

To collect events from a single object only, link that object to the EventCollector, and set the attribute HierarchicalCollection to False:

```
link event-collector-id target-object-id
modify event-collector-id HierarchicalCollection=False
commit
```

To collect events from an object and all items below it in the hierarchy, link the top object to the EventCollector object and set the HierarchicalCollection attribute to True.

```
link event-collector-id target-object-id
modify event-collector-id HierarchicalCollection=True
commit
```

If any event from any object in the entire EOM is required, simply set the attributes RootCollection and HierarchicalCollection to True.

```
modify event-collector-id RootCollection=True HierarchicalCollection=True
commit
```

Note that you cannot set the RootCollection to be True while an object linked to the EventCollector object or CollectionPoint is non-zero.

The scope can be further narrowed down by restricting it by device role and/or interface role. To do this, simply link any DeviceRole and/or InterfaceRole object to the EventCollector.

```
link event-collector-id [role-id]
commit
```

Once the subscription is set up, any events occurring that match one of the EventCollector's criteria will be sent as specified in the EventSubscription's DeliveryDetails attribute.

Event Filters

EventFilter objects can be created as children of EventCollector objects, in order to filter on faults and/or object attributes. You do not need to set the name of an EventFilter object as it will be named automatically according to the attributes set.

The two types of EventFilter objects, EventFilterFaultMask and EventFilterAttributeChange, work in similar ways.

EventFilterFaultMask

The EventFilterFaultMask allows you to set up a filter to report on specific categories of faults or individual faults only. When you create the EventFaultFilterMask object you must set the Permit attribute to True or False and the FaultCategory attribute to to match the type of fault to be reported. To report a single fault only, set the FaultCategory to "SingleFaultCode", and set the FaultCode attribute to the relevant fault number (see "EventFilterFaultMask Object").

```
setPath event-collector-id
create EventFilterFaultMask:"" Permit={true|false}
FaultCategory=fault-category FaultCode=fault-code
commit
```

EventFilterAttributeChange

The EventFilterFaultMask allows you to set up a filter to report on a specific attribute change. You must set the Permit attribute to True or False, and the AttributeName attribute to the name of the attribute to be reported. the attribute can be any attribute of any object within the EOM.

```
setPath event-collector-id
create EventFilterAttributeChange:"" Permit={true|false}
AttributeName=attribute-name
commit
```

Example

In this example, an EventSubscription object will be created and configured to send Upgraded SNMP Traps to an IP address. Two EventCollector objects will be created as children of the EventSubscription. One will catch all faults; the other will catch attribute changes of any objects in the "Customer One" hierarchy.

It assumes that initially no EventSubscription objects exist.

EventSubscription

The EventSubscription object is created as a child of /System:"System":

```
setPath /System:"System"
create EventSubscription:"Subscription 1"
commit
```

The DeliveryType attribute is set to UpgradedSNMPTrap, and the DeliveryDetails should be set to "10.100.100.10, 162, 1" which signifies that the traps should be sent to 10.100.100.10, on port 162, and version 1 SNMP traps should be sent:

```
modify EventSubscription:"Subscription 1" DeliveryType=UpgradedSnmpTrap
DeliveryDetails="10.100.100.10,162,1"
```

```
commit
```

EventCollector

The EventCollector objects are created as children of the EventSubscription object:

```
setPath EventSubscription:"Subscription 1"
create EventCollector:"Fault Collector"
create EventCollector:"Attribute Changes Collector"
commit
```

Firstly, configure the event collector that collects faults, which collects all faults in the system:

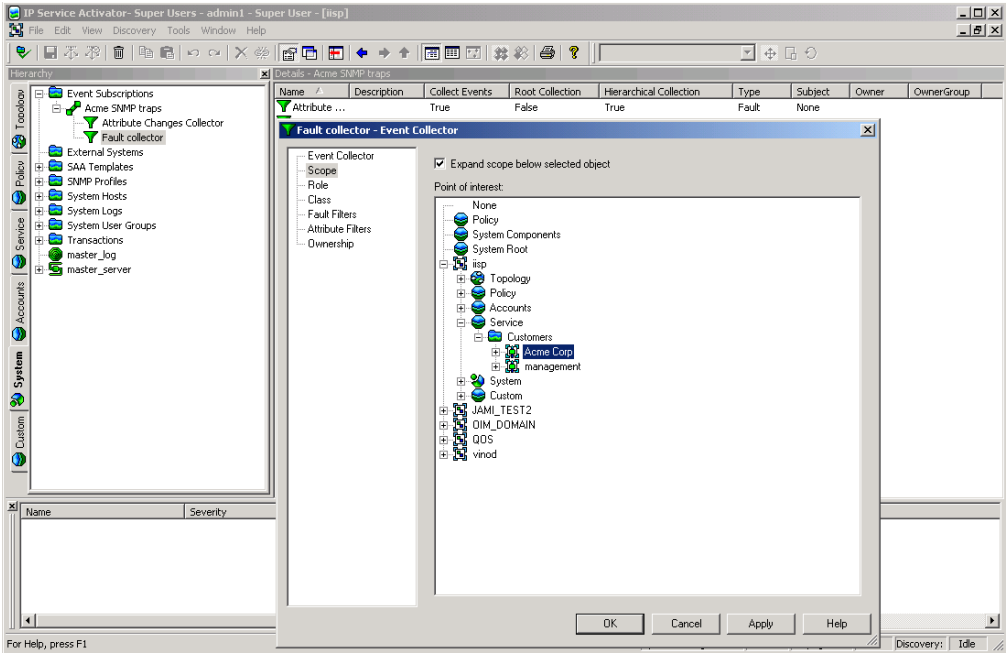
```
modify EventCollector:"Fault Collector" Type=Fault
modify EventCollector:"Fault Collector" RootCollection=True
modify EventCollector:"Fault Collector" HierarchicalCollection=True
commit
```

Then configure an event collector to collect attribute changes, which collects events for the ACME Corp customer (ID=3012) and its children:

```
modify EventCollector:"Attribute Changes Collector" Classname=<Class-name>
modify EventCollector:"Attribute Changes Collector" Type=AttributeChange
modify EventCollector:"Attribute Changes Collector" HierarchicalCollection=True
link EventCollector:"Attribute Changes Collector" [3012]
commit
```

Figure 5–14 shows that an EventSubscription object and two EventCollector objects have been created. The dialog box shows that the "Attribute Changes" EventCollector object's scope is defined as the Acme Corp customer.

Figure 5–14 EventCollector Dialog Box



Applying Parameter Sets

This example describes how to apply parameter sets to configuration targets.

Pre-requisites

- Devices, interfaces, sub-interfaces, and VC endpoints must have already been discovered, and configured with roles assigned to them.

Command Syntax

ParameterSets are applied using the **use** command. The syntax is:

```
use object-id parameter-set-id [attribute-list]
```

The above command will create a ParameterSetInstance object that is a child of both the object and the parameter set specified. The instance will also be inherited by all applicable objects, but the ParameterSetInstance object will only appear as a child of the specified object the ParameterSet object has been "used" with.

The attributes of the ParameterSetInstance to be created can be set by supplying optional attributes, or the ParameterSetInstance object can be modified after creation.

If a CollectorParameterSet is being applied, then the relevant ExternalSystem object should be linked to the created ParameterSetInstance:

```
link parameter-set-instance-id external-system-id
```

To remove the ParameterSetInstance, delete the ParameterSetInstance object:

```
delete object-id parameter-set-id
commit
```

Example

In this example, a Measurement ParameterSetInstance object (ID=15) will be applied to a Network object (ID=25) with one device. The example shows how the parameters are applied to all the interfaces belonging to the device:

```
use [25][15] Name="MeasurementInstance 1" Levels="4" OCH_MeasureCBQoS="True"
commit
```

The **use** command sets the **Levels** attribute of the ParameterSetInstance to 4. Levels is a bitwise attribute, and 4 means Interface, so the parameters will only be applied to interfaces.

The ParameterSetInstance object can be seen as a child of the Network object:

```
getchildren[25]
[701] Device:"A2504-2"
[1400] ParameterSetInstance:"Measurement Instance 1"
```

Device roles and interface roles can then be linked to the ParameterSetInstance object to indicate which interfaces the parameters should be applied to. For this example a device role of "Any", and an interface role of "Any" are linked to the ParameterSetInstance:

```
link [1400] /Policy:"Policy"/RoleDevice:"Any"
link [1400] /Policy:"Policy"/RoleInterface:"Any"
```

Using the **getParameters** command on one of the interfaces that is a child of the device (ID=701) will list the parameters from the ParameterSetInstance:

```
getchildren [701]
[702]Interface:"Ethernet0"
[705]Interface:"Serial0"
```

```
[714]Interface:"Serial1"
[715]Interface:"Null0"
[716]Interface:"Loopback0"
[740]RuleClassification:"ca1"
[5]RoleDevice:"Gateway"
getparameters [702]
r-- Boolean OCH_MeasureCBQoS = True
r-- Boolean OCH_MeasureCarQoSMIB=False
r-- Boolean OCH_MeasureMIB2Stats=False
r-- Boolean OCH_NetflowEnabled=False
r-- U32 OCH_NetflowVersion=1
r-- U32 OCH_NetflowAggregation=0
r-- U32 OCH_NetflowCacheSize=0
r-- U32 OCH_NetflowTimeoutActive=0
r-- U32 OCH_NetflowTimeoutInactive=0
```

Using the **getTargets** command on the ParameterSetInstance will display all the interface children of the device, as long as they have a role assigned.

```
gettargets [1400]
[702]Interface:"Ethernet0"
[705]Interface:"Serial0"
[714]Interface:"Serial1"
[715]Interface:"Null0"
[716]Interface:"Loopback0"
[740]RuleClassification:"ca1"
```

Applying a Collector ParameterSet is similar, except that ExternalSystem object should be linked to the ParameterSetInstance object after creation to provide the values for the parameters.

Provisioning SAA

This section describes how to create an SAA template with a Jitter operation and apply it to a VPN.

Creating an SAA template involves the following steps:

- Create an SAATemplate object
- Create the required SAAOperation objects as children of the template
- Link the SAA template object to an existing VPN

Creating an SAA Template

An SAATemplate must be created as a child of the Policy object:

```
setPath /Policy:Policy
create SAATemplate:"JitterTemplate" DeviceBits=9 TosBits=3
TypeBits=2
commit
```

> **Note:** Note that the combined values of the DeviceBits, TosBits and TypeBits attributes must be exactly 14. For more information, see "SAATemplate Object".

Creating an SAA Operation

Only one SAA operation per operation type can be created as a child of an SAATemplate object.

The name of an SAAOperation object is derived from its Type attribute value. When creating an SAAOperation object, the Name attribute must be left as an empty string ("") and the Type attribute must be specified:

```
setpath /Policy:Policy/SAATemplate:JitterTemplate
create SAAOperation:"" Type=TcpConnect
modify SAAOperation:"TcpConnect" Period=120 Timeout=25
modify SAAOperation:"TcpConnect" EnableErrorChecking=True
commit
```

Applying an SAA Template to a VPN

An SAA template is applied by linking it as a child to a VPN:

```
setpath /Policy:Policy
link Domain:"Global"/Customer:"Acme"/Vpn:"Acme VPN" SAATemplate:"JitterTemplate"
commit
```

An SAATemplate object may be linked to multiple VPNs.

Determining when Transactions Have Been Completed

It is useful to be able to determine when a transaction has been completed and to find out if configuration has been successfully applied to a device.

You can determine that a transaction has been completed by examining its attributes. You can also examine the *OPERATIONS* attribute of the Transaction object. The Transactionl object displays the concretes related to the operations.

Transaction Attributes

Activation state - It is related to every individual concrete involved in the transaction. It indicates whether the transaction's configuration changes, corresponding to each individual concrete, have actually been activated on the network.

Provisioning state - Once a transaction is committed, the ProvisioningStatus and ReasonForFailure attributes get updated. This reflects the status of the entire transaction. ProvisioningStatus can have the following status:

- Succeeded - activation status of all concretes has been set to Succeed.
- Failed - activation status of one or more concretes has been set to failed.
- Timedout - activation status of one or more concretes has been updated.

Example

/>cat [4479]

rcw	String	Description	= "3 actions"
rcw	String	Name	= "Login Banner"
r--	U32	NumberOf Concretes	= 1
rc-	Operations	OPERATIONS	=

CID: 4478 Status: Pending Operation: Create PClass: PName: PId:

CID: 4478 Status: Pending Operation: Link PClass: GenericRule PName: Login Banner PId: 4477

CID: 4478 Status: Pending Operation: Link PClass: Interface PName: 3803 PId: 3803

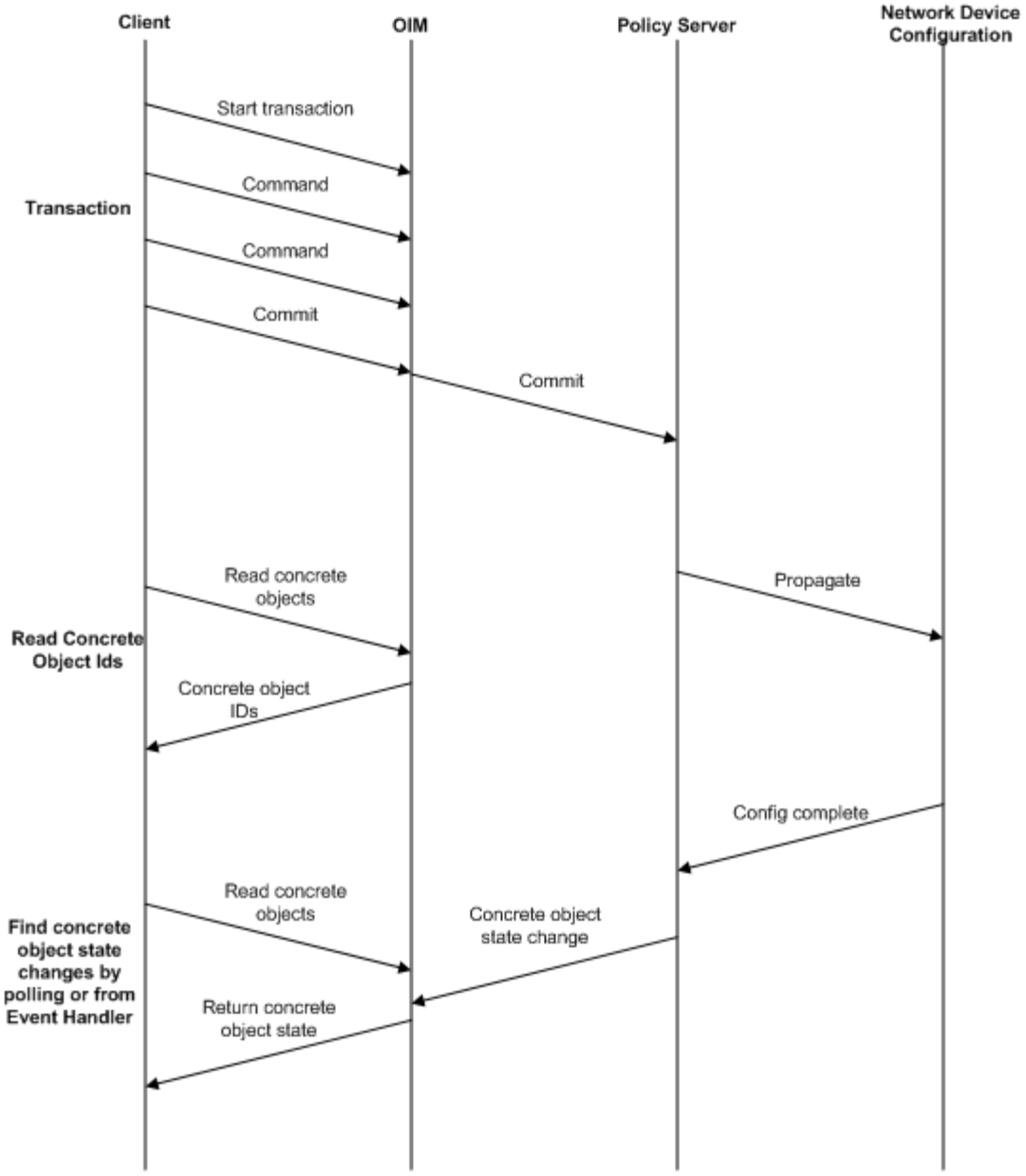
rcw	Enum	ProvisioningStatus	= Pending
rcw	String	ReasonForFailure	“”
rc-	Boolean	RollbackOn Failure	= False
rcw	Date	Schedule	= 2009/02/17 11:30:47
rc-	Enum	State	= Committed
r--	String	Username	= ‘admin’
r--	U32	ID	= 4479

In the above example an attribute called *ProvisioningStatus* displays the provisioning status of the transaction. The “status” field of the concretes displayed under the *Operations* attribute of TransactionEntry displays ‘ActivationState’ of every individual concrete.

Determining When Transactions Have Been Completed Using Concrete Objects Status

Figure 5–15 shows the workflow of a transaction to update device configuration:

Figure 5–15 Update Device Configuration Workflow



When a transaction is committed, the policy server propagates the new configuration and the device drivers or Network Processors attempt to apply it to the devices. A concrete object is created for each implementation of the rule, PHB group, VPN or Driver Script to the network. For example, if a rule is applied, a separate concrete rule is created on each interface where it is to be implemented.

To find out whether the configuration has been successfully installed you need to check the State attribute of all the concrete objects involved in the transaction. The State attribute is initially "Inactive" and is updated when the device is configured:

- A state of "Installed" means that the transaction is complete and the configuration has been successfully propagated to the device.

- A state of "Failed" means the transaction is complete and the configuration could not be applied.
- Other states indicate that the transaction is still being processed. Note that states "Finished", "RunFailed", and "RunOnceFailed" only apply to DriverScripts.

In order to determine whether configuration has been applied:

1. Locate the relevant concrete objects.
2. Check for changes in status.

Finding Relevant Concrete Objects

To find the instances where an object has been applied, you can search for all the concrete objects descended from the parent object (rule, VPN, etc), and all the concrete objects descended from the object the rule is applied to (network, customer, etc). This can be done using the command:

```
find objectID "ConcreteObject:*"
```

The intersection of the two sequences will give the set of concrete objects to examine. See "The Find Command".

An alternative method is to use the events command on the transaction that applied to the parent object. This will give a list of operations performed in the transaction, including creating, linking, or modifying concrete objects. This can be used with the subscribe command to hide operations on other objects. See "The Events Command" and "The Subscribe/Unsubscribe Commands".

Checking Concrete Object Status

There are two methods for checking concrete objects:

- Poll the concrete objects, monitoring the State attributes until the value is either "Installed" or "Failed".
- Set up an event subscription to collect AttributeChange events on the relevant concrete objects. See "Creating an Event Subscription" for more details.

Example

The following pseudo-code illustrates a script for a rule being applied to a VPN. Similar processing could be used for other scenarios:

```
link vpnID ruleID;commit
vpnConcretes = find vpnID "ConcreteObject:*"
ruleConcretes = find ruleID "ConcreteObject:*"
activeConcrete = intersection(vpnConcretes, ruleConcretes)
done = false
While not done:
 Processing = 0
 Installed = 0
 Failed = 0
 For concreteObj in activeConcretes:
  getAttributes concreteObj
  get State attribute from the returned list
  If State == "Installed":
 Installed++
Else if State == "Failed":
 Failed++
```

```
Else:
 Processing++
If Processing != 0:
 done = true
Do something with the Installed & Failed counts...
```

Managing System Users

The commands in this example illustrate how to view all users and user groups, how to add, delete, or modify a user, and how to change a user's password.

SystemUserGroups cannot be created using the OSS Integration Manager (OIM). You must use the graphical user interface to create a new SystemUserGroup.

Viewing All Users and User Groups

Super User access is required to view all users and user groups.

1. Log in to the OSS Integration Manager (OIM) using either the OIM CLI or a python OIM script.

   For example: run integration_manager_cli

2. To view all user groups and their IDs, enter:

   ```
   >find / SystemUserGroup:"*"
    [30] SystemUserGroup:"Super Users"
   ```

where 30 is the ID of the Super Users group

To view all users and their IDs, enter:

```
>find / SystemUser:"*"
  [33] SystemUser:"admin"
```

where 33 is the ID of user admin

```
  [2249] SystemUser:"jlastname"
```

Adding a User

Super User access is required to add a user.

1. Log in to OIM.
2. Find the ID of the desired SystemUserGroup.
3. Create the new user in the desired SystemUserGroup:

   ```
   >setpath <[ID]>
   ```

   where ID is the desired SystemUserGroup ID

   ```
   >create SystemUser:"<alastname>" password="<password>"
   >commit
   ```

Modifying a User

Users can change their own password and view their own attributes. The Super User can change any user's password, and view the attributes of any user.

To change a user's password:

1. Log in to OIM.

2. Find the ID of the desired SystemUser:

```
>find / SystemUser:"jl*"
```

3. Modify the password:

```
>modify [ID] password="password"
```

where ID is the desired SystemUser ID

```
>commit
```

To view user attributes:

1. View the desired user's attributes:

```
>getattributes [2249]      where ID=2249 is the desired SystemUser ID
```

r--	U32	Active	= 0
rcw	Boolean	Concurrent	= False
rcw	Boolean	Enabled	= True
r--	Date	ExpireDate	= 2004/06/16 04:59:59
r--	U32	FailedLogins	= 0
rcw	Secure	Name	= jlastname
rcw	String	Password	= "*****"
rcw	U32	PasswordExpires	= 0
rcw	String	Remarks	= ""
rcw	Boolean	ResetPassword	= False
r--	U32	Id	= 2249

Explanation: **r--** indicates a read-only attribute, while **rcw** indicates a modifiable attribute. For additional explanation of attributes, refer to "DeviceType Object".

Deleting a User

Super User access is required to delete a user.

1. Log in to OIM.
2. Find the ID of the desired SystemUser.
3. Delete the user:

```
>delete [2249]
```

where ID=2249 is the desired SystemUser ID

```
>commit
    [2256] TransactionEntry:"20040528164422"
```

Configuration Thresholding Feature: Modifying the Regular Expression

When applying Configuration Thresholding to pushing configuration commands to cartridge-driven devices, you can use this procedure to help you specify the regular expression. The provisioning of the regular expression cannot be done through the GUI, only through the OSS Integration Manager.

The Configuration Thresholding feature checks the commands to be pushed against a provisioned regular expression (**MatchesPatternTransactionSize**), counts the number of matches, and verifies that the count does not exceed the value of **MaxTransactionSize**. If the limit is exceeded, the commands are not sent and errors are raised against the concretes and the device. The regular expression and the count limit can be provisioned at a network or a device level. Configuration Thresholding is applied per device, per device transaction.

A different expression is used for different vendor devices to achieve the same purpose:

Cisco Regular Expression

For Cisco, you can count, for example, the number of "no" occurrences at the beginning of commands, to limit the amount of configuration removed in one transaction.

Huawei Regular Expression

For Huawei cartridge units, you can achieve the same function by counting the number of "undo" occurrences at the beginning of commands.

Juniper Regular Expression

For Juniper, the whole XML text (which is one large command) is considered, and the system counts the number of times that the regular expression is found in it. On JuniperXml, the system counts occurrences of "delete="delete"" attributes anywhere in the XML text.

In most cases, it gives the same results as for other vendor cartridges, except if you use "greedy" regular expressions. A "greedy" expression on Juniper can potentially match a big part of the XML text (several or many commands). On other cartridges, it only matches to a whole command at most for each count.

Limitation

A general limitation of the feature is that counting occurrences of "no" or "undo" or "delete="delete"" may not give an accurate idea of how much configuration will be removed, because if the match occurs at the top of a large code hierarchy, all code under it would be removed.

In the following sample of device code, if the command to be pushed is "no Interface 3", then MatchesPatternTransactionSize="^no" counts one removal command. But because of the code hierarchy, three commands would actually be removed from the device:

```
Interface 3
Speed
PhysicalAddress
```

An inaccurate count is particularly likely on JuniperXml code.

General Rrocedure for Changing the Regular Expression

1. Start a command line session and find the desired network or device objectID.
2. Modify the regular expression (regex) as follows:

   ```
   modify [objectID] MatchesPatternTransactionSize="<reg_expression>"
   ```

```
commit
```

3. Check if the regular expression is accepted:

```
cat
```

excerpt from response:

rcw String MatchesPatternTransactionSize = “<reg_expression>”

Specific Examples

The default global regular expression checks for “no” at the beginning of each command line. You can change it at the device level with a specific expression.

To count removal commands on a Huawei device, search for “undo” at the front of each command:

```
modify [objectID] MatchesPatternTransactionSize=”^undo”
```

To count removal commands on a Juniper device, search for “delete=”delete”” on the command line:

```
modify [objectID] MatchesPatternTransactionSize=”delete=\”delete\””
```

At the network level, you can use a regular expression that counts removal commands for several device types:

```
modify [objectID] MatchesPatternTransactionSize=”(^no|^undo|delete=\”delete\”)”
```

Applying Configuration Policies Using the OIM

To directly specify XML content for RuleGeneric policy types, and HTML and XSD schema for user-defined policy types, specify the location of the file with the required content. Alternatively, you can follow the steps specified in the examples below.

Note: While creating PolicyType or RuleGeneric objects:

- Double quotes (") should always be preceeded by an escape sequence (\).
- The OIM cannot process HTML or XML Schema languages. To insert a new line, provide an extra "\n".

HTML and XML content in the schema is given as a single continuous line in the OIM. A newline "\n" at the end of each line specifies the end of the line.

Creating a Configuration Policy Using Existing PolicyTypes

For this example, the existing configuration policy "Banner" is used.

The following objects are used:

- Device with OID [4402]
- RoleInterface:Access [11]
- RoleDevice : Gateway [5]
- PolicyType : Banner [1962]

- RuleGeneric : Exexec [8902]

1. Start by applying appropriate roles for devices and interfaces.

```
/Policy:"Policy"> link [4002] [5]
/Policy:"Policy"> link [4005] [11]
/Policy:"Policy"> commit
```

2. Create a RuleGeneric object of PolicyType Banners at the Device level.
   - Go to the Device level. RuleGeneric must have one of Vpn, Site, Device, SAP, Domain, SubInterface, Network, Interface, VcEndPointAtm, VcEndPointFr, EthernetVlan, VlanInterface, Customer, Tls, SiteL2 as a parent.

     ```
     /Policy:"Policy">  cd  [4002]
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     >
     ```
   - Create a RuleGeneric object.

     ```
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     > create RuleGeneric:"Exexec" ContentType="banners"
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     > commit
     ```

   - Provide values for all the attributes of the above-created RuleGeneric object by specifying XML content directly to the ContentValue attribute:

     ```
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     >  find / RuleGeneric:"Exexec"
     [8902] RuleGeneric:"Exexec"
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     > modify [8902] ContentValue="
     <banners xmlns=\"http://www.metasolv.com/serviceactivator/banner\">\n
      <banner>\n
       <bannerType>motd</bannerType>\n
       <message>$MOTD banner$</message>\n
      </banner>\n
     </banners>\n"
     ```

     Specify the file reference (path of the file) to the ContentValue attribute.

     ```
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     > modify [8902] ContentValue="D:\motdbanner.xml"
     ```

     The XML File reference (D:\motdbanner.xml) will contain pure XML content as given below.

     ```
     <banners xmlns="http://www.metasolv.com/serviceactivator/banner">
      <banner>
       <bannerType>motd</bannerType>
       <message>$MOTD Banner$</message>
      </banner>
     </banners>
     /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com"
     > commit
     ```

     Note: Make sure that the XML content given in the ContentValue attribute is well defined and valid.

3. Link the Role Interface and Role Device to the RuleGeneric object:

```
/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com">
link [8902] [11]
/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com">
link [8902] [5]
/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"rot7206-5.oracle.com">com
mit
```

The Configuration Policy Exexec created is applied to the device and the concretes are created.

Creating User-defined PolicyTypes

You can create user-defined PolicyTypes in the following ways:

- by specifying HTML and XSD schema content directly to OIM as a string.
- by loading the **userdefinedpolicy.policy** file to the Domain that has references to the HTML and XSD schema files.

Using the HTML and XSD Schema Content

In this example a user-defined policy type called staticNats for NetworkAddressTranslation is created.

1. Navigate to the folder where the user-defined policy can be stored. On the **Policy** tab, expand **Service**, then select **General**. Or, execute the following command:

```
/>cd [1900]
    /Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"
```

2. Review the existing PolicyTypes that are available.

```
/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">ls
    [1901] PolicyType:"Banner"
    [1902] PolicyType:"IP Pools"
    [1903] PolicyType:"Key Chain"
    [1904] PolicyType:"Prefix List"
    [1905] PolicyType:"SNMP Community"
    [1906] PolicyType:"SNMP Host"
    [1907] PolicyType:"Static Route"
    [1908] PolicyType:"User Authentication"
    [1909] PolicyType:"User Data"
```

3. Create PolicyType "staticnats" using the OIM **Create** command.

```
/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">create
PolicyType:"statisnats"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">commit
    [3060] TransactionEntry:"20090306101813"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">ls
    [1901] PolicyType:"Banner"
    [3059] PolicyType:"statisnats"
    [1902] PolicyType:"IP Pools"
    [1903] PolicyType:"Key Chain"
    [1904] PolicyType:"Prefix List"
    [1905] PolicyType:"SNMP Community"
    [1906] PolicyType:"SNMP Host"
    [1907] PolicyType:"Static Route"
    [1908] PolicyType:"User Authentication"
    [1909] PolicyType:"User Data"
```

```
/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">cat
[3059]
    rcw Enum            Category            = Service
    rcw Boolean         Disabled            = False
    rcw String          HtmlPage            = ""
    rcw String          Name                = "statisnats"
    rcw String          Remarks             = ""
    rcw String          Schema              = ""
    rcw String          Type                = ""
    rcw Boolean         Visible             = True
    r-- U32             Id                  = 3059
```

4. Provide the HTML content of the user-defined PolicyType.

```
/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">modify
PolicyType:"statisnats" HtmlPage="
<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 3.2//EN\">\n
<html>\n
<head><title>com.metasolv.serviceactivator.staticnat.StaticNatsDocument</title>
\n
<link rel=\"stylesheet\" type=\"text/css\" href=\"style.css\"></link>\n
<style TYPE=\"text/css\">\n
<!--\n
TABLE, TD, TR, BODY\n
{\n
font-family: 'ms sans serif', sans-serif;\n
font-size:9pt;\n
background-color: threedface;\n
}\n
INPUT, SELECT, BUTTON\n
{\n
font-family: 'ms sans serif', sans-serif;\n
font-size:9pt;\n}\n
-->\n
</style>\n
<script LANGUAGE=\"JavaScript\" SRC=\"DynamicForm.js\"></script
>\n
<script LANGUAGE=\"JavaScript\" SRC=\"ValidateForm.js\"></script>\n
\n
</head>\n
<body BGCOLOR=\"#ECE9D8\" onload=\"doOnLoad()\">\n
\n
<script language=\"JavaScript\">\n
\n
displayNameByNamePut(\"staticNats.staticNat.outsideIpAddress\", \"Outside IP
Address\");\n
displayNameByNamePut(\"staticNats.staticNat.insideIpAddress\", \"Inside IP
Address\");\n
\n
</script>\n
\n
<form name=\"mainform\" action=\"\" method=\"POST\">\n
<input type=\"hidden\" name=\"staticNats.xmlns\"
value=\"http://www.metasolv.com/serviceactivator/staticnat\"/>
\n
<table valign=\"top\" border=\"0\" width=\"100%\">\n
<tr id=\"staticNats.staticNat.add\">\n
<td>\n
\n
</td>\n
<td>\n
<button onclick=\"
```

```
addRows('staticNats.staticNat', 'null')\">Add Static Nat</button>\n
</td>\n
\n
</tr>\n
\n
</table>\n
<script language=\"JavaScript\">\n
var error_string = '';\nfunction validateFormElements(form) {\n
var success = true;\n
for(var i=0;i < form.elements.length;i++) {\n
        key = form.elements[i];\n
        var keyname = key.name;\n
        if (typeof keyname == \"undefined\"){\n
           continue;\n        }\n
        keyname = getPureCid(key
name);\n
    if (keyname == \"staticNats.staticNat.outsideIpAddress\") {\n
        if (! validate_string('Outside IP Address', key.value,
0,999999999,'','',false)) {\n
             success = false;\n
        }\n
    }\n
    if (
keyname == \"staticNats.staticNat.insideIpAddress\") {\n
if (! validate_string('Inside IP Address', key.value, 0,999999999,'','',false))
{\n
             success = false;\n
        }\n
    }\n
    }\n
    if (!success) {\n
  alert(error_string);\n
        error_string='';\n
    }\n
    return success;\n
}\n
</script>\n
</form>\n
<form name=\"templateForm\" action=\"\" method=\"POST\">\n
<table valign=\"top\" border=\"0\" width=\"100%\" class=\"HIDDEN\">\n
<tr id=\"staticNats.staticNat.t\">\n
<td colspan=\"2\">\n
<fieldset><legend>Static Nat</legend>\n
<table valign=\"top\" border=\"0\" width=\"100%\">\n
<tr>\n
<td valign=\"top\">\n
<table valign=\"top\" border=\"0\" width=\"100%\" class=\"top\">\n
<tr id=\"staticNats.staticNat.insideIpAddress.t\">\n
<td valign=\"top\">\n
Inside IP Address\n
</td>\n
<td valign=\"top\">\n<input type=\"text\"
name=\"staticNats.staticNat.insideIpAddress.t\" size=\"0\"/>\n
</td>\n
\n
</tr>\n
<tr id=\"staticNats.staticNat.outsideIpAddress.t\">\n
<td valign=\"top\">\n
Outside IP Address\n
</td>\n
```

```
<td valign=\"top\">\n
<input type=\"text\" name=\"staticNats.staticNat.outsideIpAddress.t\" size=\"0\
"/>\n
</td>\n
\n
</tr>\n
\n
</table>\n
\n
</td>\n
\n
</tr>\n
\n
</table>\n
\n
</fieldset>\n
\n
</td>\n
<td valign=\"top\">\n
<button onclick=\"deleteRow('staticNats.staticNat')\">Delete</button>\n
</td>\n
\n
</tr>\n
\n
</table>\n
\n
</form>\n
<script LANGUAGE=\"JavaScript\">\nfunction doOnLoad() {\n
}\n
</script>\n
</body>\n
</html>\n"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">commit
    [3061] TransactionEntry:"20090306102049"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">modify
PolicyType:"statisnats" Schema="
<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n
\n
<!-- Copyright (c) 2007,2008 Oracle. All rights reserved. Oracle is a trademark
of  -->\n
<!-- Oracle Corporation and/or its affiliates. Other names may be trademarks
-->\n
<!-- or their respective owners.
-->\n
\n
<xs:schema
targetNamespace=\"http://www.metasolv.com/serviceactivator/staticnat\" \n
xmlns:lib=\"http://www.metasolv.com/serviceactivator/devicemodel\" \n
xmlns:app=\"http://www.metasolv.com/serviceactivator/guiextensions\" \n
xmlns:ipsa=\"http://www.metasolv.com/serviceactivator/documentation\"\n
xmlns:sn=\"http://www.metasolv.com/serviceactivator/staticnat\" \n
xmlns:xs=\"http://www.w3.org/2001/XMLSchema\" elementFormDefault=\"qualified\"
att
ributeFormDefault=\"unqualified\">\n
    <xs:import
namespace=\"http://www.metasolv.com/serviceactivator/devicemodel\"
schemaLocation=\"../networkprocessor/devicemodel/datatypes.xsd\"/>\n
<xs:complexType name=\"StaticNat\">\n
```

```
   <xs:sequence>\n
      <xs:element name=\"insideIpAddress\" type=\"lib:IPv4Address\">\n
         <xs:annotation>\n
            <xs:documentation>Inside IPaddress</xs:documentation>\n
            <xs:appinfo>\n
               <app:gui>\n
                  <app:displayName>Inside IP Address</app:displayName>\n
               </app:gui>\n
            </xs:appinfo>\n
         </xs:annotation>\n
      </xs:element>\n
      <xs:element name=\"outsideIpAddress\" type=\"lib:IPv4Address\">\n
         <xs:annotation>\n
            <xs:documentation>Outside IP address</xs:documentation>\n
            <xs:appinfo>\n
               <app:gui>\n
                  <app:displayName>Outside IP Address</app:displayName>\n
               </app:gui>\n
            </xs:appinfo>\n
         </xs:annotation>\n
      </xs:element>\n
   </xs:sequence>\n
</xs:complexType>\n
<xs:complexType name=\"StaticNats\">\n
   <xs:sequence>\n
      <xs:element name=\"staticNat\" type=\"sn:StaticNat\" minOccurs=\"0\"
maxOccurs=\"unbounded\">\n
         <xs:annotation>\n
            <xs:appinfo>\n
               <app:gui>\n
                  <app:displayName>Static Nat</app:displayName>\n
               </app:gui>\n
            </xs:appinfo>\n
         </xs:annotation>\n
      </xs:element>\n
   </xs:sequence>\n
</xs:complexType>\n
<xs:element name=\"staticNats\" type=\"sn:StaticNats\">\n
   <xs:annotation>\n
      <xs:documentation>This configuration policy allows you to create
static Network Address Translation on a router to change the IP address
that is routed in or out of the router and help conserve IP
addresses.</xs:documentation>\n
      <xs:appinfo>\n
         <ipsa:documentation>\n
            <ipsa:seealso>Creating Device Level Configuration Policies
            </ipsa:seealso>\n
            <ipsa:since>5.2.0</ipsa:since>\n
            <ipsa:usagecontext>Device</ipsa:usagecontext>\n
            <ipsa:cardinality>Multiple static [no default]Ts (Network
Address Translation) can be defined in the same policy, or across
multiple policies on the same device.</ipsa:cardinality>\n
         </ipsa:documentation>\n
      </xs:appinfo>\n
   </xs:annotation>\n
</xs:element>\n
</xs:schema>\n"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General">commit
    [3062] TransactionEntry:"20090306102733"
```

5. Provide a value for the **Type** attribute. This will be treated as content-type for RuleGeneric.

```
/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"/PolicyTy
pe:"statisnats">modify [3059] Type="statisNats"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"/PolicyTy
pe:"statisnats">commit
    [3063] TransactionEntry:"20090306105107"
```

6. Create a RuleGeneric object using the newly created PolicyType (here staticNats).

```
/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"/PolicyTy
pe:"statisnats">find / Device:"*"
    [2592] Device:"NY01.oracle.com"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"/PolicyTy
pe:"statisnats">cd [2592]
    /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com"

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">create
RuleGeneric:"statex1" ContentType="staticNats" ContentValue="
<staticNats xmlns=\"http://www.metasolv.com/serviceactivator/staticnat\">\n
<staticNat>\n
<insideIpAddress>1.2.2.1</insideIpAddress>\n
<outsideIpAddress>2.1.1.1</outsideIpAddress>\n
</staticNat>\n
</staticNats>\n"

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">commit
    [3089] TransactionEntry:"20090306114456"

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">link
[3086] [11]

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">commit
    [3091] TransactionEntry:"20090306114658"

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">link
[3086] [5]

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">commit
    [3093] TransactionEntry:"20090306114721"
```

Using the User-defined .policy File

This example creates a user-defined policy type called staticNats for Network Address Translation.

Pre-requisites:

- User defined configuration policy source files must be available at appropriate directories. For more information see *IP Service Activator SDK Configuration Policy Extension Developer Guide*.
- For every Configuration Policy, three files are required.
  - The **ConfigPolicy.policy** file - to load the policy to a Domain (for example, **staticnat.policy**)
  - HTML file - to provide an input GUI (for example, **staticNats.html**)
  - XSD file - schema for the configuration policy (for example, **staticNat.xsd**)

Note: Make sure that all these files are available on the server in the IPSA_HOME/SamplePolicy directory.

- You must create the **.policy** file.

  See the line specified in **staticnat.policy** file to load the user-defined staticNat configuration policy to the domain. The line is:

  ```
  Service:Custom|;Static [no default]T;staticNats;staticNats;staticNat;0;1;
  ```

  The information provided in the **.policy** file has the following fields:

  – Service:Custom - Directory structure to load the policy to GUI.

  – Static [no default]T - Name of the user defined configuration policy to be displayed on the GUI.

  – staticNats - Type of the configuration policy.

  – staticNats - Name of the html file.

  – StaticNat - Name of the xsd file.

1. Load the **.policy** file to the domain.

   Available domains are listed:

   ```
   />find / Domain:"*"
    [584] Domain:"test"
    [1973] Domain:"test1"
   ```

   Modify the LoadPolicy attribute of the domain and specify the name of the **ConfigPolicy.policy** file (**staticNat.policy** in this example.)

   ```
   />cd [584]
       /Policy:"Policy"/Domain:"test"

   /Policy:"Policy"/Domain:"test">cat
       rcw U32             AllowAsIn            = 1
       rcw Boolean         AsOverride           = True
       rcw String          BgpMd5Key            = "******"
       rcw Boolean         DomainAsOverride     = True
       rcw String          LoadPolicy           = "unset"
       rcw String          Name                 = "test"
       rcw Boolean         PeCeSendExtendedCommunity = False
       rcw Boolean         PeCeSendStandardCommunity = False
       rcw Boolean         PePeSendStandardCommunity = True
       rcw String          Remarks              = ""
       rcw String          UseLoopback          = "0"
       rcw U32             VrfTableLimit        = 0
       rcw U32             VrfTableLimitWarning = 75
       r-- U32             Id                   = 584
   /Policy:"Policy"/Domain:"test">modify LoadPolicy="staticNat.policy"

   /Policy:"Policy"/Domain:"test">commit
       [3385] TransactionEntry:"20090320065237"
   ```

2. Create the RuleGeneric object using the newly created PolicyType (staticNats in this example.)

   ```
   /Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"/PolicyTy
   pe:"statisnats">find / Device:"*"
   ```

```
    [2592] Device:"NY01.oracle.com"

/Policy:"Policy"/PolicyTypeFolder:"Service"/PolicyTypeFolder:"General"/PolicyTy
pe:"statisnats">cd [2592]
    /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com"

/Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">create
RuleGeneric:"statex1" ContentType="staticNats" ContentValue="
<staticNats xmlns=\"http://www.metasolv.com/serviceactivator/staticnat\">\n
<staticNat>\n
<insideIpAddress>1.2.2.1</insideIpAddress>\n
<outsideIpAddress>2.1.1.1</outsideIpAddress>\n
</staticNat>\n
</staticNats>\n"
```

"ContentValue" can take the path of the xml file.

- Create an XML file (here **staticnat.xml**) with all the values of the configuration policy and make sure that the file is valid against the schema (here **staticNat.xsd**).
- Provide the path of the XML file to ContentValue attribute. OIM will parse the file and take care of escaping special characters (" to \") and new line (\n).

  For example:

  ```
  /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">create
  RuleGeneric:"statex1" ContentType="staticNats"
  ContentValue="export\OracleCommunication\ServiceActivator\xmls\staticnat.xml"
  ```

  See the XML content of the **staticnat.xml** file as shown that does not have escape sequence characters and \n:

  ```
  <staticNats xmlns="http://www.metasolv.com/serviceactivator/staticnat">
  <staticNat>
  <insideIpAddress>1.1.1.1</insideIpAddress>
  <outsideIpAddress>2.2.2.2</outsideIpAddress>
  </staticNat>
  </staticNats>

  /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">commit
      [3089] TransactionEntry:"20090306114456"

  /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">link
  [3086] [11]

  /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">commit
      [3091] TransactionEntry:"20090306114658"

  /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">link
  [3086] [5]

  /Policy:"Policy"/Domain:"test"/Network:"test"/Device:"NY01.oracle.com">commit
      [3093] TransactionEntry:"20090306114721"
  ```

6

Error Handling

This chapter provides details about error handling.

Exceptions

The API returns four general exception types. Specific information concerning the cause of the exception is contained in a descriptive string parameter called *Reason*.

CommandSyntaxException

This exception type occurs if the command syntax is not recognized.

- The command issued is unknown
- The path entered is malformed
- The attribute entered is unknown
- The command is malformed and has not been recognized
- Unexpected regular expression
- The attribute appears more than once in the parameter list
- The parameter supplied is not valid in this context

CommandExecutionError

Indicates that a command could not execute. The returned string is used to indicate the specific error condition:

- The command is invalid
- The system could not resolve the object supplied to an existing or a valid object
- A required parameter is missing from the argument list
- Too many parameters to the command
- The command requires a path
- The command requires precisely two path arguments
- The attribute value specified is out of range
- The specified EOM object type is invalid and cannot be created
- Cannot change the value for this attribute
- Command cannot be applied on this object or you do not have permission to perform this command

- Command has resulted in Object Model integrity violation
- The find command requires at least one Object, Attribute value or Regular Expression
- The find command does not accept a Regular Expression and an Object simultaneously
- This object is not a valid child of the specified class
- This object cannot be unlinked from the specified object
- This object cannot be linked to the specified object
- This object cannot be copied
- The copied object cannot be of the type specified
- The attribute values of the Global Template are read-only
- The type of the object is read-only and cannot be modified
- The link to a Global Template and the local values cannot be changed in the same time
- No child matching the given attributes could be found
- An attempt to merge a TransactionEntry has failed
- An attempt to rollback a TransactionEntry has failed
- The user does not have permissions to perform this command

OimSystemException

A general system error has occurred. The returned string is used to indicate the specific error condition. Examples are:

- Failure to communicate with the Policy Server
- Memory Error
- Shutdown is in progress
- This command cannot be executed

LoginException

Indicates errors occurring whilst trying to login to the API. The returned string is used to indicate the specific error condition:

- Invalid user name or password
- The login procedure has repeatedly failed

A

Command Grammar

This appendix provides full details of the command grammar.

Basic Grammar for Commands

The basic grammar for commands is described here using Regular Expressions:

token* means zero to many appearances of the token

token+ means at least one appearance of the token

token? means zero or one appearance of the token

(tokenA | tokenB) means either tokenA or tokenB

```
command ::= COMMAND_[no default]ME (' ' parameter )*
```

> **Note:** The command name is one of the following:
>
> abort | alias | commit | copy | create | delete | discover | events | getAccess | find | findParameters | getAttributes | getChildren | getID | getName | getPath | getParents | gettargets | link | listTrans | login | logout | manage | merge | modify | rollback | schedule | setPath | subscribe | unlink | unsubscribe | unmanage | use | unuse | xmlexport

```
parameter ::= relative_path | absolute_path | attribute_couple | reg_exp
absolute_path ::= '/'  relative_path*
relative_path ::=  object  ('/' object )*
object ::= prefixed_name | object_id | ".." | '.'
object_id ::= '[' digit+ ']'
prefixed_name ::= EOM_TYPE ':' object_name
```

The EOM Type strings recognized are the objects accessible in the EOM. See "The External Object Model" for full details.

```
object_name ::= string | letter+
attribute_couple ::= ATTRIBUTE_[no default]ME  '='  attribute_value
Attribute names are defined in the OIM External Object Model Reference.
attribute_value ::=  string |  (letter | digit | '.' | '-' | '_' )+
regexp ::= '"' str_content * ( '?' | '*' ) str_content* '"'
string ::= '"' str_content * '"'
str_content::= <any non escaped character> | "\\" | "\?" | "\n" | "\*" | "\""
```

Note: The quote characters may appear inside a string if it is escaped with a backslash. The backslash character in a quoted string needs to be escaped with a further backslash. For example:

Domain:"Some domain"

Domain:"A domain with \"embedded\" quote marks"

```
digit ::= [0-9]
letter ::= [a-z] | [A-Z]
```

Each command follows the same basic command format. For example, a command string using the **unmanage** command would look as follows:

```
unmanage domain:"Europe"/site:"London"/device:"UK120"
```

This example unmanages device 'UK120' within domain 'europe' and site 'London'.

B

Command Correlation

This appendix summarizes the commands that you can perform on each object in the EOM.

Command Correlation Table

The following table indicates the commands that you can perform on each object in the EOM.

Table B–1 Command Correlation Table

Object	create delete	modify	link unlink	copy	use unuse	discover	manage unmanage	merge rollback
Account	No	No	Yes	No	No	No	No	No
Classification	Yes	Yes	Yes	No	No	No	No	No
ClassificationGroup	Yes	Yes	Yes	No	No	No	No	No
Component	No	No	No	No	No	No	No	No
ConcreteObject	No	No	No	No	No	No	No	No
CoS	Yes	Yes	No	No	No	No	No	No
Customer	Yes	Yes	Yes	No	No	No	No	No
Device	Yes	Yes	Yes	No	No	Yes	Yes	No
DeviceType	Yes	Yes	No	No	No	No	No	No
Domain	No	No	Yes	No	No	No	No	No
DriverScript	No	No	Yes	No	No	No	No	No
EthernetVlan	Yes	Yes	Yes	No	No	No	No	No
EventCollector	Yes	Yes	Yes	No	No	No	No	No
EventFilter	Yes	Yes	Yes	No	No	No	No	No
EventSubscription	Yes	Yes	Yes	No	No	No	No	No
ExternalSystem**	Yes	Yes	Yes	No	No	No	No	No
Fault	No	No	No	No	No	No	No	No
Interface	Yes	No	Yes	No	No	No	No	No
Network	Yes	Yes	Yes	No	No	No	No	No

Table B–1 (Cont.) Command Correlation Table

Object	create delete	modify	link unlink	copy	use unuse	discover	manage unmanage	merge rollback
Options	No	Yes	No	No	No	No	No	No
PacketMarking	Yes	Yes	Yes	No	No	No	No	No
ParameterSet*	No	No	No	No	Yes	No	No	No
ParameterSetInstance**	No	Yes	No	No	No	No	No	No
Period	No	No	No	No	No	No	No	No
PHB	Yes	Yes	Yes	No	No	No	No	No
PHBAtm	Yes	Yes	Yes	No	No	No	No	No
PHBFrts	Yes	Yes	Yes	No	No	No	No	No
PHBGroup	No	No	No	No	Yes	No	No	No
PHBGroupInstance	No	Yes	No	No	No	No	No	No
PHBGroupMqc*	Yes	Yes	Yes	No	Yes	No	No	No
PHBMqc	Yes	Yes	Yes	No	No	No	No	No
PHBPolicingAction	Yes	Yes	Yes	No	No	No	No	No
Policy	No	No	No	No	No	No	No	No
RoleDevice	Yes	No	Yes	No	No	No	No	No
RoleInterface	Yes	No	Yes	No	No	No	No	No
Root	No	No	No	No	No	No	No	No
RtNumber	Yes	Yes	Yes	No	No	No	No	No
RuleAccess	Yes	Yes	Yes	Yes	No	No	No	No
RuleClassification	Yes	Yes	Yes	Yes	No	No	No	No
RulePolicing	Yes	Yes	Yes	Yes	No	No	No	No
SAATemplate	No	Yes	Yes	No	No	No	No	No
SAAOperation	Yes	Yes	Yes	No	No	No	No	No
Segment	Yes	No	Yes	No	No	Yes	No	No
Site	Yes	Yes	Yes	No	No	No	No	No
SiteHub	Yes	No	Yes	No	No	No	No	No
SiteL2	Yes	Yes	Yes	No	No	No	No	No
StaticRoute	Yes	Yes	Yes	No	No	No	No	No
SubLayer	Yes	No	Yes	No	No	No	No	No
SubInterface	Yes	No	Yes	No	No	No	No	No
System	No	No	No	No	No	No	No	No
Tls	Yes	Yes	Yes	No	No	No	No	No
Topology	No	No	No	No	No	No	No	No
TrafficApplication	No	No	Yes	No	No	No	No	No
TrafficCompound	No	No	Yes	No	No	No	No	No

Table B–1 (Cont.) Command Correlation Table

Object	create delete	modify	link unlink	copy	use unuse	discover	manage unmanage	merge rollback
TrafficDomainName	No	No	Yes	No	No	No	No	No
TrafficGroup	No	No	Yes	No	No	No	No	No
TrafficMime	No	No	Yes	No	No	No	No	No
TrafficPacketMarking	No	No	Yes	No	No	No	No	No
TrafficPort	No	No	Yes	No	No	No	No	No
TrafficSubApplication	No	No	Yes	No	No	No	No	No
TrafficURL	Yes	No	Yes	No	No	No	No	No
TransactionEntry	No	No	No	No	No	No	No	Yes
VcEndpointAtm	Yes	Yes	Yes	No	No	No	No	No
VcEndpointFr	Yes	Yes	Yes	No	No	No	No	No
VlanInterface	Yes	Yes	Yes	No	No	No	No	No
Vpn	Yes	Yes	Yes	No	No	No	No	No

C

Sample IP Service Activator Transaction Publisher Class

This appendix illustrates how to use Oracle Communications IP Service Activator External Object Model (EOM) events that are generated by IP Service Activator OSS Integration Manager (OIM) on TransactionEntry objects. The sample OSS Java Development Library (OJDL) code, shown on "Sample OJDL file" and explained with examples "Coding examples", transforms the EOM events into Java Message Service (JMS) events, which are then published on a JMS topic.

The events are meant to be used by an external JMS-enabled application to track the activation status of its IP Service Activator transactions. Activation status information in IP Service Activator is available for all transactions through TransactionEntry's ProvisioningStatus and ReasonForFailure attributes.

Changes to these two attributes, as well as TransactionEntry's State attribute, are encoded in the JMS messages and can be easily decoded by the external client. The format of the messages can also be customized.

Types of JMS Events

Two types of JMS events are generated: TransactionCreation and AttributeChange. These events are described in the following sections.

TransactionCreation

This event is generated when a transaction is created by an IP Service Activator client. The transaction for the generated create event is treated as active in the sample. All active transactions are stored in a container, so that subsequent events on them can be analyzed and translated into AttributeChange events.

AttributeChange

This event is generated when one or more attributes of an active transaction change. The event indicates that one of the following has occurred:

- Transaction's State attribute changed. This is an indication that a scheduled transaction has been merged into the IP Service Activator object model and its activation has begun.
- Transaction's ProvisioningStatus attribute changed. Three cases are possible:
  - Transaction's ProvisioningStatus changed to Success. This is an indication that directly committed or scheduled transaction completed activation successfully.

- Transaction's ProvisioningStatus changed to Failed. This is an indication that directly committed or scheduled transaction has failed activation. The same event should contain new value for the ReasonForFailure attribute.
- Transaction's ProvisioningStatus changed to Timedout. This is an indication that activation status of directly committed or scheduled transaction was not confirmed during the configured timeout period.

Note:

- If AttributeChange is generated and indicates that “Transaction's State attribute changed” occurred, it is followed by another AttributeChange indicating that any one of the three cases of “Transaction's ProvisioningStatus attribute changed” occurred.
- An AttributeChange indicating any one of the three cases of “Transaction's ProvisioningStatus attribute changed” is the final event for a transaction.

Sample OJDL file

There is a code example available in the additional documentation included with the OJDL libraries. For more information, see the example “SampleTransactionStatusPublisher” in the ojdlSamples folder of the OJDL zip package. Various parts of the sample file are explained in the section "Coding examples".

Coding examples

The following examples illustrate how the code in the OJDL sample file (shown starting on "Sample OJDL file") can be used to transforms the OSS Integration Manager (OIM) events on TransactionEntry objects into Java Message Service (JMS) events.

In this section, OSS Integration Manager (OIM) events are defined as events generated by the IP Service Activator OIM. OJDL clients poll for those events by calling EomSession.getEvents.

Implementation of the java.lang.Runnable Interface

Implements the run() method of Runnable. The run() method executes an infinite loop that is only interrupted when the application exits. It repeatedly calls

```
getEomEvents()
```

The returned events are then processed:

```
for (int i = 0; i < events.length; i++) {
System.out.println(events[i]);
processEomEvent( events[i]);
}
```

The run method pauses before repeating the same cycle:

```
Thread.sleep( IPSA_EVENT_POLLING_INTERVAL);
```

A real transaction publisher will be executing the run method on a separate thread concurrently with other activities.

EOMSession Management

An EOM session is used in order to obtain OIM events:

```
events = m_eomSession.getEvents(false);
```

The session is reestablished when exceptions occur (for example due to IP connectivity loss):

```
// reconnection attempt
System.err.println("Re-attempting to establish EOM session to   host: " + IPSA_
HOST + " port: " + IPSA_PORT + ".  Attempt
 # " + numAttempts);
openEomSession();
```

Up to [IPSA_NUM_RECONNECT_ATTEMPTS] attempts are made to reestablish a failed session

```
for( int numAttempts = 0; numAttempts <= IPSA_NUM_RECONNECT_ATTEMPTS;
++numAttempts) {
...
 System.err.println("Re-attempting to establish EOM
 session to host: " + IPSA_HOST +
  " port: " + IPSA_PORT + ". Attempt # " + numAttempts);
 openEomSession();
...
}
```

Handling of Create Events

Create events on TransactionEntry objects are recognized using a pattern matcher and processed in the onCreateEvent() method:

```
...
Matcher createEomEventMatcher =
 createEomEventPattern.matcher( eomEvent);
...
if( createEomEventMatcher.find()) {
 onCreateEvent( createEomEventMatcher);
}
```

The onCreateEvent () method stores a record of the transaction being created (referred to as active transaction):

```
m_activeTransactions.put( infoObject.Id, infoObject);
```

Next, it formats the text of the outgoing JMS message and calls publishJmsMessage().

Handling of Modify Events

Modify events on active transactions are recognized using a pattern matcher and processed in the onModifyEvent() method:

```
Matcher modifyEomEventMatcher =
 modifyEomEventPattern.matcher( eomEvent);
...
else if( modifyEomEventMatcher.find()) {
 onModifyEvent( modifyEomEventMatcher, eomEvent);
}
```

The onModifyEvent method makes sure that the event is on an active transaction before processing it:

```
if( m_activeTransactions.containsKey( objectId)) {
 ...
}
```

The onModifyEvent method uses four patterns and matchers to match changes on the following attributes: ScheduleTime, State, ReasonForFailure, and ProvisioningStatus.

In addition to the new value, the old value of the attribute is retrieved from the stored info object. Both values are used to format an attribute change clause in the text of the outgoing JMS event. More than one attribute change clause can occur in the same JMS event. This would typically be the case when the event indicates Provisioning Status changed to failed. The same event will contain ReasonForFailure.

All attribute change clauses are concatenated to form the text of the outgoing JMS message passed to publishJmsMessage().

Customizing the Format of the JMS Message

The sample uses templates for the text of both TransactionCreate:

```
protected final String createEventTemplate =
"TransactionCreation(\n" +
"Id: \"%Id%\",\n" +
"Name: \"%Name%\",\n" +
"NumberOfConcretes: \"%NumberOfConcretes%\",\n" +         "ScheduleTime:
\"%ScheduleTime%\",\n" +
"State: \"%State%\"\n" +
")\n";
```

and AttributeChange events:

```
protected final String attributeChangeEventTemplate =
"AttributeChangeEvent(\n" +
"Id: \"%Id%\",\n" +
"Name: \"%Name%\",\n" +
"%AttributeChanges%)\n";
```

There is also a template for a single attribute change:

```
protected final String attributeChangeTemplate =
"%AttributeName%: ( \"%OldValue%\", \"%NewValue%\")\n";
```

These templates contain markers in the format %markername% which are replaced with the actual values. The templates can be changed to any desired format. For example, you can change the templates so that the messages conform to a particular XML schema or DTD.

Publishing the Outgoing Message

The JMS message is published from within the publishJmsMessage:

```
TextMessage message = m_jmsSession.createTextMessage();
message.setText(messageText);
m_jmsPublisher.publish(message);
```

The sample uses a JMS topic to publish the message. This may be changed to a queue if the receiver is using a queue.

Initializing the JMS Entities

All JMS initialization is done in the initializeJms() method. This method may require customization to suit the specific application needs.