



## CHAPTER 6

# MPLS Provisioning

---

The service provider's backbone is comprised of the core provider edge (PE) device and its provider routers. An MPLS VPN consists of a set of customer sites that are interconnected through an MPLS provider core network. At each site, there are one or more customer edge (CE) devices, which attach to one or more PEs.

The Cisco IP Solution Center (ISC) provisioning engine for MPLS accesses the configuration files on both the CE and PE to compute the necessary changes to the configuration files to support the service on the PE to CE link (or PE to CLE, PE to MVRFCE to CE, or PE to MVRFCE to CLE).

This chapter describes MPLS VPN service concepts and the steps required to provision MPLS VPN services using the ISC API. The provisioning example includes the process flow from creating the inventory to auditing the service deployment.

For information on MPLS provisioning using the ISC GUI, see the [Cisco IP Solution Center MPLS VPN User Guide, 5.0](#).

This chapter contains the following sections:

- [MPLS Service Definitions, page 6-2](#)
- [MPLS Service Requests, page 6-6](#)
- [End-to-End Provisioning Process, page 6-7](#)

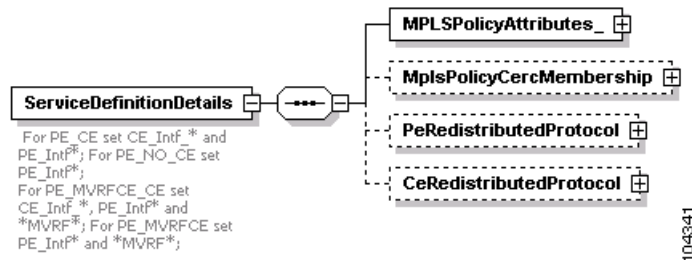
# MPLS Service Definitions

An MPLS service definition defines attributes for the policy type (**MPLSPolicyAttributes**), and can include the CE routing community (CERC) membership and redistributed protocol information.

CERC membership defines the CE routing community for this policy and is represented by the VPN routing and forwarding tables (VRFs), and the redistributed protocols define the metric attributes. MPLS policy attributes are described in the [MPLS Policy Attributes](#) section.

Figure 6-1 shows the schema diagram for an MPLS service definition.

**Figure 6-1** MPLS Service Definition Schema Diagram



For each service definition property, you can set an additional attribute, **editable=true**, to allow the network operator to override these attributes when creating the service request. If an attribute is set to **editable=false**, these attributes cannot be changed in the service request.



## Note

When a property has the additional attribute **editable=true**, all the related and child attributes are also editable.

See the following example:

```
<objectPath xsi:type="ns1:CIMObjectPath">
  <className xsi:type="xsd:string">ServiceDefinitionDetails</className>
  <properties xsi:type="ns1:CIMPropertyList"
    soapenc:arrayType="ns1:CIMProperty[]">
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">SubType</name>
      <value xsi:type="xsd:string">PE_CE</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">PE_Intf_Type</name>
      <value xsi:type="xsd:string">GigabitEthernet</value>
      <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">true</value>
      </qualifier>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">PE_Intf_Desc</name>
      <value xsi:type="xsd:string">Interface Description</value>
      <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">true</value>
      </qualifier>
    </item>
  </properties>
</objectPath>
```

```

<name xsi:type="xsd:string">PE_Intf_Shutdown</name>
<value xsi:type="xsd:string">TRUE</value>
<qualifier xsi:type="nsl:CIMQualifier">
  <name xsi:type="xsd:string">editable</name>
  <value xsi:type="xsd:string">true</value>
</qualifier>
</item>

```

## MPLS Policy Attributes

The MPLS policy attributes include; the policy subtype, device interfaces and encapsulation types, IP addressing schemes, routing protocols, and VPN membership information. These values are set based on the policy subtype.

The following MPLS policy subtypes are supported:

- PE\_CE—A standard MPLS policy for the PE\_CE link. This is the default MPLS policy for ISC.
- PE\_NO\_CE—In this policy type, you specify only the PE interface information. The CE device at the customer location is not managed by ISC.
- PE\_MVRFCE\_CE—A Multi-VPN routing and forwarding CE (MVRFCE) network has multiple CE devices that connect to one MVRFCE device. The MVRFCE stores the VRFs for all VPNs in the customer network and connects directly with the PE device at the edge of the provider network. ISC manages the PE to MVRFCE to CE link.
- PE\_MVRFCE\_NO\_CE—ISC manages the PE to MVRFCE link. The CE device at the customer site is not managed by ISC.

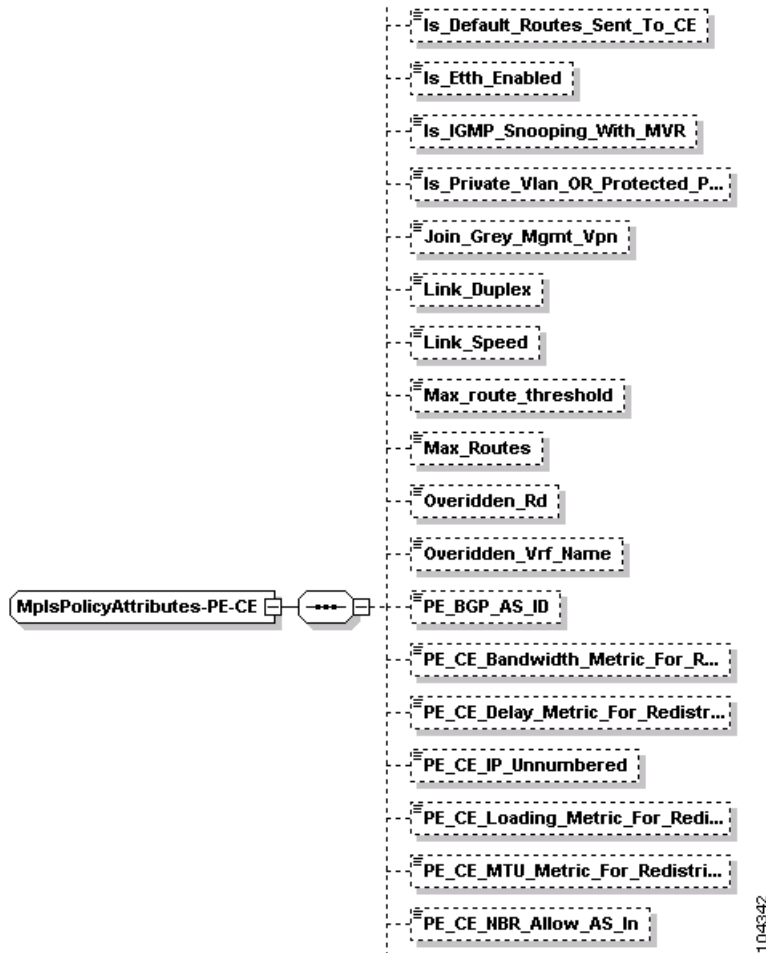


### Note

For all policy subtypes with no CE present, you do not declare the CE devices to be CPEs, and you do not set policy attributes for the CE devices in the service definition.

There are numerous properties that can be set for an MPLS policy. [Figure 6-2](#) shows a partial schema diagram for the **MPLSPolicAttributes** with **SubType=PE\_CE**.

Figure 6-2 MPLS Policy Attributes Schema Diagram



The following XML example shows a partial list of the properties that can be specified for the **MplsPolicyAttributes**:

```
<objectPath xsi:type="ns1:CIMObjectPath">
  <className xsi:type="xsd:string">ServiceDefinitionDetails</className>
  <properties xsi:type="ns1:CIMPropertyList"
    soapenc:arrayType="ns1:CIMProperty[]">
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">SubType</name>
      <value xsi:type="xsd:string">PE_CE</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">PE_CE_IP_Unnumbered</name>
      <value xsi:type="xsd:string">>false</value>
      <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
      </qualifier>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">IGMP_Query_Time</name>
      <value xsi:type="xsd:string">25</value>
      <qualifier xsi:type="ns1:CIMQualifier">
```

```

        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">IGMP_Mode</name>
    <value xsi:type="xsd:string">COMPATIBLE</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Overridden_Vrf_Name</name>
    <value xsi:type="xsd:string">vrf1</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Autopick_Vlan_ID</name>
    <value xsi:type="xsd:string">true</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Extra_CE_Loopback_Required</name>
    <value xsi:type="xsd:string">FALSE</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Auto_Assign_IP_Address</name>
    <value xsi:type="xsd:string">TRUE</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">IP_Address_pool_type</name>
    <value xsi:type="xsd:string">Region Pool</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">PE_CE_Routing_Protocol</name>
    <value xsi:type="xsd:string">RIP</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">>true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Is_Default_Routes_Sent_To_CE</name>
    <value xsi:type="xsd:string">FALSE</value>
    <qualifier xsi:type="ns1:CIMQualifier">

```

```

        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Redistribute_Static</name>
    <value xsi:type="xsd:string">TRUE</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Redistribute_Connected</name>
    <value xsi:type="xsd:string">TRUE</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">true</value>
    </qualifier>
</item>
<item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">Max_Routes</name>
    <value xsi:type="xsd:string">10000</value>
    <qualifier xsi:type="ns1:CIMQualifier">
        <name xsi:type="xsd:string">editable</name>
        <value xsi:type="xsd:string">true</value>
    </qualifier>
</item>

```

## MPLS Service Requests

An MPLS service request specifies the MPLS policy (service definition) to use, the list of links, referred to as MPLS VPN links, and the link attributes. Use the link attribute settings in the service request to override any policy settings defined as editable in the service definition.

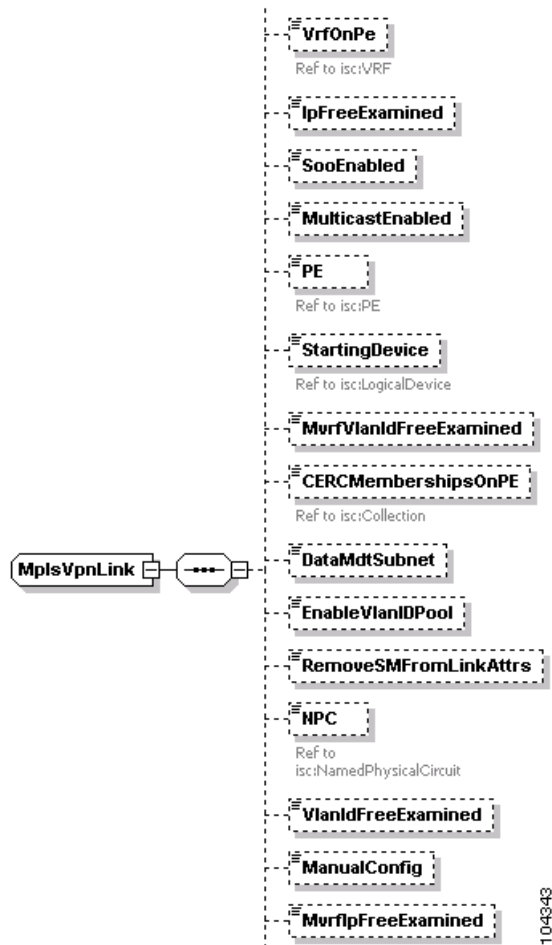


### Note

You can also integrate an ISC template with an MPLS service request and associate one or more templates to the CPE and PE devices. See [Chapter 4, “Using Templates,”](#) for more information.

ISC supports an extensive list of properties that can be set for MPLS VPN links. [Figure 6-3](#) shows a partial schema diagram for the **MplsVpnLink** in a service request.

Figure 6-3 MPLS VPN Link Schema Diagram



## End-to-End Provisioning Process

The following sections describe the required steps for using the API to provision MPLS VPNs, and include the operation, class, and required parameters for each step.

### Process Summary

MPLS provisioning using the API includes the following end-to-end operations:

- Step 1** Prior to creating the inventory, you need to do the following:
- Send a login XML request to generate a session ID. This is used each time you access the system. (see [Session](#), page 3-7)
  - View the status of the API by executing

```
runNbi x $ISC_HOME/resources/nbi/xml/examples/Session/Login.xml
```

If the API is running, you receive a session token. (see [Checking the API Status, page 2-1](#))

## Step 2 Create Inventory

- **CreateProvider.xml**—create a provider;
- **CreateRegion.xml**—create a region;
- **CreateOrganization.xml**—create customer organisation;
- **CreateSite.xml**—create customer site;
- **CreateCiscoRouter.xml**—create devices (Cisco IOS router);
- **CreateCat.xml**—create devices (Catalyst switch);
- **CreatePE.xml**—declare a device as **PE** and assign it to regions;
- **CreateCpe.xml**—declare a device as **CPE** and assign it to sites.

For more information, see [Create Inventory, page 6-9](#).

## Step 3 Create Resource Pools

Create a **VPN** and select a **CERC**:

- **CreateRouteTarget.xml**—informs PEs which routes should be inserted into the appropriate VRFs;
- **CreateRouteDistinguisher.xml**—helps the CE router advertise IP subnets to the PE routers. The RD value must be a globally unique value to avoid conflict with other prefixes;
- **CreateCERC.xml**—create a CERC;
- **CreateVPN.xml**—create a VPN.

For more information, see [Create Resource Pools, page 6-11](#).

## Step 4 Collect Device Configurations

- **CreateTaskServiceOrderCollection.xml**—upload the current configuration from the device to the ISC database.

For more information, see [Collect Device Configurations, page 6-12](#).

## Step 5 Create an MPLS Policy

- **CreateMPLSServiceDefn\_PE\_CE.xml**—define a policy template for a PE-CE link;
- **CreateMPLSServiceDefn\_PE\_NO\_CE.xml**—define a policy template for a PE-no CE link;
- **CreateMPLSServiceDefn\_MVRF\_CE.xml**—define a policy template for a MVRF-CE link;
- **CreateMPLSServiceDefn\_MVRF\_NO\_CE.xml**—define a policy template for a MVRF-no CE link.

For more information, see [Create an MPLS Policy, page 6-14](#).

## Step 6 Create an MPLS Service Request




---

**Note** To modify a service request, see [Modifying a Service Request, page 1-14](#). To delete or purge a service request, you need the returned Locator ID. (See [Step 7](#).) For examples of modify and delete operations, go to examples.tar under the MPLS directory.

---

Define which service definition to use, the **MplsVpnLink**, and the link attributes:

- **CreateMPLSServiceOrder\_PE\_CE.xml**—create service request using the PE-CE policy;
- **CreateMPLSServiceOrder\_PE\_NO\_CE.xml**—create service request using the PE-noCE policy;

- **CreateMPLSServiceOrder\_MVFR\_CE.xml**—create service request using the MVFR-CE policy;
- **CreateMPLSServiceOrder\_MVRF\_NO\_CE.xml**—create service request using the MVRF-no CE policy.

For more information, see [Creating an MPLS Service Request, page 6-14](#).

**Step 7** When you submit a service order XML request, ISC returns a **LocatorId** in the XML response.

Make a record of the locator ID or service name for all service orders and service requests. The locator ID is required to view a service order, to perform a service order task (configuration audit or functional audit), and for all subsequent requests related to the service order or service request.

For more information, see [Service Order Response, page 2-12](#).

**Step 8** Using the LocatorId, you can view the log output for the created and deployed service request. This includes all messages according to the level specified in the properties file.

For more information, see [Viewing Task Logs, page 5-23](#).

## Detailed Process and Attributes

This MPLS provisioning example includes the following operations:

- [Create Inventory](#)
- [Create Resource Pools](#)
- [Collect Device Configurations](#)
- [Create an MPLS Policy](#)
- [Creating an MPLS Service Request](#)

This section provides an example provisioning process using XML examples. The inventory of XML examples for the ISC API is available here:

[http://www.cisco.com/en/US/docs/net\\_mgmt/ip\\_solution\\_center/5.0/developer/reference/xmlapi.zip](http://www.cisco.com/en/US/docs/net_mgmt/ip_solution_center/5.0/developer/reference/xmlapi.zip)

## Create Inventory

**Step 1** Create a **Provider** and **Region**.

The provider is the administrative domain of an Internet service provider (ISP), with one border gateway protocol (BGP) autonomous system (AS) number. The network owned by the provider is called the backbone network. If an ISP has two AS numbers, you must define it as two provider administrative domains (PADs). Each provider can contain multiple regions.

**Table 6-1** Create Provider and Region

Operation	className	Required Keywords
createInstance	Provider	<ul style="list-style-type: none"> <li>• Name</li> <li>• AsNumber</li> </ul>
	Region	<ul style="list-style-type: none"> <li>• Name</li> <li>• Provider</li> </ul>

**XML Examples:**

- CreateProvider.xml
- CreateRegion.xml

**Step 2** Create a Customer (**Organization**) and **Site**.

**Table 6-2** Create Customer and Site

Operation	className	Required Keywords
createInstance	Organization	<ul style="list-style-type: none"> <li>• Name</li> </ul>
	Site	<ul style="list-style-type: none"> <li>• Name</li> <li>• Organization</li> </ul>

**XML Examples:**

- CreateOrganization.xml
- CreateSite.xml

**Step 3** Create devices.

In most cases, devices are Cisco IOS routers and Catalyst switches.

**Table 6-3** Create Devices

Operation	className	Required Keywords
createInstance	<ul style="list-style-type: none"> <li>• CiscoRouter</li> <li>• CatOS</li> </ul>	One or more of the following: <ul style="list-style-type: none"> <li>• ManagementIPAddress</li> <li>• HostName</li> <li>• DomainName</li> </ul>

**XML Examples:**

- CreateCiscoRouter.xml
- CreateCat.xml

**Step 4** Declare devices as **PEs** and assign them to regions.

**Table 6-4** Create PEs

Operation	className	Required Keywords
createInstance	PE	<ul style="list-style-type: none"> <li>• Provider</li> <li>• Region</li> <li>• Role= <ul style="list-style-type: none"> <li>– N-PE</li> <li>– U-PE</li> <li>– P</li> <li>– PE-AGG</li> </ul> </li> <li>• Device</li> <li>• Interface</li> </ul>

**XML Example:**

- CreatePE.xml

**Step 5** Declare devices as **Cpes** and assign them to sites.

When you declare a device as a **Cpe**, you also specify a **ManagementType** and interface information.

**Table 6-5** Create CPEs

Operation	className	Required Keywords
createInstance	Cpe	<ul style="list-style-type: none"> <li>• Site</li> <li>• Device</li> <li>• ManagementType</li> </ul>

**XML Example:**

- CreateCpe.xml

## Create Resource Pools

For MPLS provisioning, you define route targets, route distinguishers, CERCs, and VPNs.

A route target informs PEs which routes should be inserted into the appropriate VRFs. Every VPN route is tagged with one or more route targets when it is exported from a VRF and offered to other VRFs.

The IP subnets advertised by the CE routers to the PE routers are augmented with route distinguishers (RDs). The RD value must be a globally unique value to avoid conflict with other prefixes.

**Table 6-6** Create Resource Pools

Operation	classNames	Required Keywords
createInstance	<ul style="list-style-type: none"> <li>RouteTarget</li> <li>RouteDistinguisher</li> </ul>	<ul style="list-style-type: none"> <li>Start</li> <li>Size</li> <li>AssocClassType</li> <li>AssocClassId</li> </ul>

Create a **VPN** and select a **CERC**.

**Table 6-7** Create VPNs and CERCs

Operation	className	Required Parameters
createInstance	<ul style="list-style-type: none"> <li>VPN</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> <li>CERC</li> </ul> <p><i>or</i></p> <ul style="list-style-type: none"> <li>CreateDefaultCERC</li> </ul>
	<ul style="list-style-type: none"> <li>CERC</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> <li>Provider</li> <li>SpokeRouteTarget</li> <li>HubRouteTarget</li> </ul>

A CERC can either be full mesh or hub and spoke. If you specify hub and spoke, you must specify both the **SpokeRouteTarget** and **HubRouteTarget**. For a full mesh CERC, only **SpokeRouteTarget** is required.

**XML Examples:**

- CreateRouteTarget.xml
- CreateRouteDistinguisher.xml
- CreateCERC.xml
- CreateVPN.xml

## Collect Device Configurations

A device configuration collection is a task. This task uploads the current configuration from the device to the ISC database. The collection task is executed through a service request, and the service request is scheduled using a service order.

**Table 6-8 Collect Device Configurations**

Operation	className	Required Parameters
createInstance	ServiceOrder	<ul style="list-style-type: none"> <li>• ServiceName</li> <li>• NumberofRequests</li> <li>• ServiceRequest</li> </ul>
	ServiceRequest	<ul style="list-style-type: none"> <li>• RequestName</li> <li>• Type=Task</li> <li>• ServiceRequestDetails</li> </ul>
	ServiceRequestDetails	<ul style="list-style-type: none"> <li>• SubType=Collection</li> <li>• Device (or DeviceGroup)</li> </ul> <p><b>Note</b> You must select at least one device or device group.</p> <ul style="list-style-type: none"> <li>• RetrieveVersion=true</li> <li>• RetrieveDeviceInterfaces=true</li> </ul>

The following example is a partial XML request used to collect the configuration from device ensw4000-1:

```
<className xsi:type="xsd:string">ServiceRequestDetails</className>
  <properties xsi:type="ns1:CIMPropertyList"
    soapenc:arrayType="ns1:CIMProperty[]">
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">SubType</name>
      <value xsi:type="xsd:string">COLLECTION</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Device</name>
      <value xsi:type="xsd:string">ensw4000-1</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">DeviceGroup</name>
      <value xsi:type="xsd:string">PE-Group</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">RetrieveDeviceAttributes</name>
      <value xsi:type="xsd:string">true</value> </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">RetrieveDeviceInterfaces</name>
      <value xsi:type="xsd:string">true</value> </item>
  </properties>
</objectPath>
```

**XML Example:**

- CreateTaskServiceOrderCollection.xml

## Create an MPLS Policy

An MPLS service policy (defined in a service definition) is a template of the parameters needed to define a service request. Once you define the policy template, it can be used by all MPLS service requests that share a common set of attributes.

An MPLS service definition consists of the **MplsPolicyAttributes**. The **MplsPolicyAttributes** define the properties specific to the policy subtype.

**Table 6-9** Create an MPLS Policy

Operation	className	Required Parameters
createInstance	ServiceDefinition	<ul style="list-style-type: none"> <li>Name</li> <li>Type=Mpls</li> <li>ServiceDefinitionDetails</li> </ul>
	ServiceDefinitionDetails	<ul style="list-style-type: none"> <li>MPLSPolicyAttributes</li> <li>Provider or Organization</li> </ul> <p><b>Note</b> If you do not specify a <b>Provider</b> or <b>Organization</b>, the service policy becomes a global policy.</p>
	MPLSPolicyAttributes	<ul style="list-style-type: none"> <li>SubType= <ul style="list-style-type: none"> <li>PE_CE</li> <li>PE_NO_CE</li> <li>PE_MVRFCE_CE</li> <li>PE_MVRFCE_NO_CE</li> </ul> </li> </ul>

### XML Examples:

- CreateMPLSServiceDefn\_PE\_CE.xml
- CreateMPLSServiceDefn\_PE\_NO\_CE.xml
- CreateMPLSServiceDefn\_MVRF\_CE.xml
- CreateMPLSServiceDefn\_MVRF\_NO\_CE.xml

## Creating an MPLS Service Request

An MPLS service request defines the service definition to use, the **MplsVpnLink** and the link attributes. The **MplsVpnLink** specifies the device interfaces involved in this service request. The link attributes contain any policy setting overrides for properties set as editable in the service definition. A service request can specify one or more MPLS VPN links.



**Note** Use **performBatchOperation** to group the service order and service request into one XML request.



**Note** The service request name must be unique for each NBI API.

**Table 6-10** Create an MPLS Service Request

Operation	className	Required Parameters
createInstance	ServiceOrder	<ul style="list-style-type: none"> <li>• ServiceName</li> <li>• NumberOfRequests</li> <li>• ServiceRequest</li> </ul>
	ServiceRequest	<ul style="list-style-type: none"> <li>• RequestName</li> <li>• Type=Mpls</li> <li>• ServiceRequestDetails</li> </ul>
	ServiceRequestDetails	<ul style="list-style-type: none"> <li>• ServiceDefinition <ul style="list-style-type: none"> <li>– ServiceDefinitionType=Mpls</li> </ul> </li> <li>• MplsVpnLink</li> </ul>
	MplsVpnLink	<ul style="list-style-type: none"> <li>• NPC</li> <li><i>or</i></li> <li>• ManualConfig=true <ul style="list-style-type: none"> <li>– PE</li> <li>– Cpe</li> </ul> </li> </ul> <p><b>Note</b> You must specify either <b>NPC</b> or <b>ManualConfig</b> to define the interfaces for the MPLS VPN links. If you use <b>ManualConfig</b>, you must also specify the interfaces (for example, <b>CE_Intf_Name</b> and <b>PE_Intf_Name</b>).</p> <ul style="list-style-type: none"> <li>• LinkTemplate (optional)</li> </ul> <p><b>Note</b> See the “<a href="#">Templates in a Service Request</a>” section on page 4-9.</p>

**Note**

The attributes **PE\_Template**, **PE\_Intf\_Template**, **CE\_Template**, and **CE\_Intf\_Template** allow NBI access to variables designed to hold template blobs (template blobs were used during MPLS provisioning in legacy versions of ISC).

**XML Examples:**

- CreateMPLSServiceOrder\_PE\_CE.xml
- CreateMPLSServiceOrder\_PE\_NO\_CE.xml
- CreateMPLSServiceOrder\_MVFR\_CE.xml
- CreateMPLSServiceOrder\_MVRF\_NO\_CE.xml

## Auditing Service Requests

A configuration audit occurs automatically each time you deploy a service request. During this configuration audit, ISC verifies that all Cisco IOS commands are present and that they have the correct syntax. An audit also verifies that there were no errors during deployment by examining the commands configured by the service request on the target devices. If the device configuration does not match what is defined in the service request, the audit flags a warning and sets the service request to a *Failed Audit* or *Lost* state.

If you do not want the configuration audit to occur, change the value for the **Audit** parameter. The **Audit** parameter supports these values:

- **Audit**—This is the default. A successfully deployed service request is automatically audited unless this flag is changed.
- **NoAudit**—Do not perform a configuration audit when the service request is deployed.
- **ForceAudit**—Perform a configuration audit even if the service request deployment is not successful.

You can use the Audit parameter with a **Create**, **Modify**, or **Decommission** service request or a **Deployment** task. See the [“Service Decommission” section on page 3-10](#) for more information. To perform a configuration audit as a separate task, or an MPLS functional audit, see the [“Tasks” section on page 3-8](#).