Common APIs

IPsec, firewall, NAT: **These features are not supported in this release**.

This chapter describes the Cisco IP Solution Center (ISC) APIs for operations that are common to all ISC services. These operations include; creating inventory in the ISC respository, session login, auditing, deployment and collection tasks, database event notifications, and general purpose XML requests.

This chapter contains the following sections:

Inventory

You can create, modify, delete, or view any inventory object in the ISC repository. Use the following API operations to:

- Create an object—createInstance
- Modify an object—modifyInstance (Resource pools do not support the *Modify* operation.)
- Delete an object—deleteInstance
- View an object—enumerateInstances

Note When you send a view request (**enumerateInstances**) for an object, the response returns the *create* and *modify* date information.

Inventory APIs are divided into the following categories:

- Devices—Physical devices in the network
- Resource Pools—IP address pools, VLAN pools, route target, route distinguisher
- Topology—CE routing communities (CERCs), virtual private networks (VPNs), and named physical circuits (NPCs)
- Groupings—Access domains, device groups, collection zones.

Note XML examples for all APIs are located at the following URL:
http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/isc/4_0/api/apiref/examples/index.htm

Devices

The inventory APIs for devices allow you to define a physical device in the ISC repository. Every network element that ISC manages must be defined as a device in the system. An element is any device from which ISC can collect configuration information.

Table 3-1 shows the devices available as inventory objects.

Table 3-1 Inventory Objects—Devices

className	Required Parameters	XML Examples
AAAServer	• Number of Retries • Address • AuthServerType= – RADIUS – NTDOMAIN – SDI – TACACS+ • Role= – AUTHENTICATION – ACCOUNTING – BOTH	• CreateAAServer.xml • CreateAAServerNTDOMAIN.xml • CreateAAServerRADIUS.xml • CreateAAServerSDI.xml • CreateAAServerTACACS.xml
CatOs (Cat OS operating system)	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreateCat.xml
CatIOS (Cisco IOS operating system)	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreateCatIOS.xml
CiscoRouter	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreateCiscoRouter.xml

Table 3-1 Inventory Objects—Devices (continued)

className	Required Parameters	XML Examples
Cpe	• Site • Device • Name	CreateCpe.xml
PE	• Device • Region	CreatePE.xml
PIX	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreatePIX.xml
TerminalServer	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreateTerminalServer.xml
VPN3000	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreateVPN3000.xml
VPNServicesModule	• Parent • SlotNumber	CreateVPNSM.xml
IE2100	One or more of the following: • IPAddress • HostName • DomainName	CreateIE2100.xml
NonCiscoDevice	One or more of the following: • ManagementIPAddress • HostName • DomainName	CreateNonCiscoDevice.xml

Resource Pools

Resource pools allow you to manage various pool types and to associate a pool with any service model object in the ISC repository. Pools help to automate service deployment.

ISC supports the following resource pools:

- IP address pools—Defined and assigned to regions.
- Multicast pools—Used for multicast MPLS VPNs.

- Route Distinguisher (RD) pool—The IP subnets advertised by the CPE devices to the PE devices are augmented with a 64-bit prefix, which is the route distinguisher.
- Route Target (RT) pool—The MPLS mechanism that informs PEs which routes to insert into the appropriate VPN routing and forwarding table (VRF). Every VPN route is tagged with one or more route targets when it is exported from a VRF and offered to other VRFs.
- Site-of-origin pool—The pool of values for the site-of-origin attribute. The site-of-origin attribute prevents routing loops when a site has multiple connections to the MPLS VPN backbone.
- VLAN ID pool—VLAN ID pools are attached to an access domain. To have ISC automatically assign VLANs to end-to-end wire links (for L2VPN provisioning), you can specify the **AutoPickVlanId** option.
- VC ID pool—A 32-bit identifier that is shared between the two PEs. The VC ID is a globally unique value.

Note The *Modify* operation is not supported with resource pools.

Table 3-2 shows the resource pools available as inventory objects.

Table 3-2 Inventory Objects—Resource Pools

className	Required Parameters	XML Examples
IPAddressPool	• IPAddressPool • SubnetMask • AssocClassType= – Region – VPN	CreateIPAddressPool.xml
MulticastAddrPool	• MulticastAddress	CreateMulticastAddrPool.xml
RouteDistinguisher	• Start • Size • AssocClassType=Provider • AssocClassId	CreateRouteDistinguisher.xml
RouteTarget	• Start • Size • AssocClassType=Provider • AssocClassId	CreateRouteTarget.xml
SiteOfOrigin	• Start • Size • AssocClassType=Provider • AssocClassId	CreateSiteOfOrigin.xml

Table 3-2 Inventory Objects—Resource Pools (continued)

className	Required Parameters	XML Examples
VcIdPool	• Start • Size	CreateVcIdPool.xml
VlanIdPool	• Start • Size • AssocClassType=Provider • AssocClassId	CreateVlanIdPool.xml

Topology

In complex topologies, it is sometimes necessary to further define the connectivity between CE and PE devices into groups, such as CE routing communities (CERCs), virtual private networks (VPNs), and named physical circuits (NPCs).

Tip Use the Topology tool in the GUI to check CERC memberships and the resulting VPNs.

Table 3-3 shows the topology elements available as ISC inventory objects.

Table 3-3 Inventory Objects—Topology

className	Required Parameters	XML Examples
CERC	• Provider	• CreateCERC.xml **Note** The CreateCERC.xml appends to any existing CERC. To create a new CERC, first run a modify CERC to remove the existing CERC attributes. • CreateVPN_DefaultCERC.xml
VPN	• Name • Organization	• CreateVPN.xml
NamedPhysicalCircuit	• PhysicalLink – SrcDevice – DestDevice – SrcIfName – DestIfName	• CreateNamedPhysicalCircuit.xml
NamedPhysicalCircutRing	• LocatorId	• CreateNamedPhysicalCircuitRing.xml • CreateNamed PhysicalCircuitRingExisting.xml

Tip Use **ViewVRF.xml** to view the VRF routing tables.

Groupings

ISC is designed to provision a large number of devices through its distributed architecture. Groupings are provided to simplify management tasks of devices for administrative or geographical reasons or for scalability.

- Access domain—The access domain object is associated with the provider access domain (PAD). Each PE-POP is assigned to an access domain.
- Device groups—Devices that are organized for collection and management purposes.
- Organization (Customer)— A requestor of VPN services from an Internet service provider (ISP).
- Site—A set of IP systems with mutual IP connectivity between them without the use of a VPN.
- Provider— An enterprise that provides internet services for customers.
- Region—A group of PE devices within a single border gateway protocol autonomous system (BGP AS).
- Collection zones—Collection servers are used to off load the work of the master server when the number of devices in the network increases. Network devices are associated with these collection servers using collection zones.
- Network objects—A network object is a group of IP addresses to use in a Firewall policy, or for traffic classification in a QoS policy, instead of specifying a single IP address.

Table 3-4 shows the groupings as inventory objects.

Table 3-4 Inventory Objects—Groupings

className	Required Parameters	XML Examples
AccessDomain	• Provider • ReservedVlanPool – Start – Size – Managed	CreateAccessDomain.xml
DeviceGroup	• Name • Devices	CreateDeviceGroup.xml
Organization	• Name • ContactInfo • SiteOfOrigin=enabled (use for multiple connections to the MPLS backbone)	CreateOrganization.xml
Site	• Name • Customer	CreateSite.xml
Provider	• Name • AsNumber	CreateProvider.xml
Region	• Name • Provider	CreateRegion.xml

Table 3-4 Inventory Objects—Groupings (continued)

className	Required Parameters	XML Examples
CollectionZone	• Name • VPNSCHost	CreateCollectionZone.xml
NetworkObject	• Name • Type= – STRING – NETWORK – HOST • Values • ContainerId • FQCN (container type) – Global – Customer – Site – Cpe	CreateNetworkObject.xml

Session

Sessions are specialized API operations. ISC supports the following session operations:

- createSession—Login
- deleteSession—Logout

The first XML request sent by the client is a login request. The login request generates a session ID, which is used each time you access the system. The session ID is valid for a configurable period of time. During this time, you can make an indefinite amount of calls to the server, using the session ID for authentication. Each call to the server resets the time-to-live (ttl) time back to the original period of time. The default session time is 20 minutes.

The API license is global to the installation and is checked at the start of each session. If the API license does not exist, the session is not granted. During the session, if a user attempts to provision a service that is not licensed, an error is returned.

Table 3-5 User Login

className	Required Parameters	XML Examples
Session	• LoginName • LoginPassword	Login.xml

The following example shows the XML response to a login request. The **SessionId** is indicated in **bold**.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://www.cisco.com/cim-cx/2.0" xmlns:ns1="urn:CIM">
  <soapenv:Header>
    <ns0:message id="87855" timestamp="2003-10-10T19:43:16.380Z"
sessiontoken="93E0A400406693604270C6B6A07731A4" />
  </soapenv:Header>
  <soapenv:Body>
    <ns1:createSessionResponse>
      <returns xsi:type="ns1:CIMPropertyList" soapenc:arrayType="ns1:CIMProperty[]">
        <item xsi:type="ns1:CIMProperty">
          <name xsi:type="xsd:string">SessionId</name>
          <value xsi:type="xsd:string">93E0A400406693604270C6B6A07731A4</value>
        </item>
      </returns>
    </ns1:createSessionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Tasks

Task APIs are used for auditing, deployment, and collection activities. Tasks follow the same service order/service request structure as other ISC services, where the service request is executed as part of a service order.

- For device-related tasks (Collection and SLA Collection), you must enter the device or device group.
- For service request-related tasks (Certificate Enrollment Audit, Configuration Audit, Service Deployment, IPsec and MPLS Functional Audits), you must enter the service request locator ID.

Tasks are scheduled using the **DesiredDueDate** field in the service request or service order. A task with a **DesiredDueDate** that is in the past, or within 5 minutes of the current time, is executed immediately. Tasks with due dates in the future are scheduled.

ISC uses the following guidelines for **DesiredDueDate**

- If there is no due date in the service request, the task uses the service order due date.
- If there is a date in the service request, the service request due date overrides the service order due date for that particular task.
- If there is no due date specified in the service order or the service request, the service is not deployed.

Tasks can be created, delete, and viewed using the API. However, you can only modify the **DesiredDueDate** field in a Task.

All Tasks are deployed using a service order that specifies a service request with **Type=Task**.

Viewing a Task

To obtain information about task service requests, run a view of the task (ViewTask.xml), using the **TaskLocatorId**.

Use this process to retrieve the **TaskLocatorId**:

Step 1 Record the service order **LocatorId** that is returned in the XML response.

When you submit a service order XML request for any task, ISC returns a **LocatorId** in the XML response. This locator ID is associated with the service order or request **Name**.

Step 2 Run a view of the service order using ViewServiceOrder.xml, and the **LocatorId** from Step 1.

Step 3 Record the **TaskLocatorId** that is returned in the XML response.

Step 4 Run a view of the Task using ViewTask.xml. Specify **className=PersistentTask** and use the **TaskLocatorId** from Step 3.

The API supports the following tasks:

- Certificate Enrollment Audit
- Collection
- Service Decommission
- Configuration Audit
- Service Deployment
- IPsec Functional Audit
- MPLS Functional Audit
- SLA Collection

Certificate Enrollment Audit

A certificate enrollment audit is performed on devices to verify the certificates issued to them by a certificate authority (CA). ISC supports certificate enrollment audits for site-to-site and remote access IPsec service requests.

A certificate enrollment audit does the following:

- Verifies the certificates issued by the CA to a device or device groups.
- Confirms the presence of the root certificate and device certificate for the certificate chain of the specified trust point.
- Returns a summary that indicates the certificate enrollment status and expiration status on the audited devices.

Note ISC supports certificate enrollment audit for Cisco IOS devices, but not for PIX and VPN3000 devices.

Table 3-6 Certificate Enrollment Task

className	Required Parameters	XML Examples
ServiceRequestDetails	• SubType= CERT_ENROLLMENTAUDIT • Device (or DeviceGroup) • TrustedPort • IncludeRootCert	CreateTaskServiceOrderCertAudit.xml

Collection

A collection operation collects configuration information from network devices and serves the following purposes:

- It loads the current configuration information for the devices that populate many of the configuration cells.
- It verifies reachability and passwords for the devices from which it collects configuration files.

Table 3-7 Collection Task

className	Required Parameters	XML Examples
ServiceRequestDetails	• SubType=COLLECTION • Device (or DeviceGroup) **Note** You must select at least one device or device group. • RetreiveVersion=true • RetreiveDeviceInterface=true	Create TaskServiceOrderCollection.xml

Tip Performing a device collection on your network is more accurate and efficient than manually entering individual device information.

Service Decommission

Decommissioning a service request removes a service from all devices associated with the service request.

Note To decommission a service request that includes templates, you must first negate the template information on the device. See the "Removing Template Configurations" section on page 4-13 for more information.

During the decommission process, ISC creates the necessary removal configuration to delete the service from each device and automatically audits the configuration to ensure that the service is completely removed. Once audited, the service request changes to a *Closed* state.

Table 3-8 Decommission Task

className	Required Parameters	XML Examples
ServiceRequestDetails	• SubType=DECOMMISSION • LocatorId **Note** The **LocatorId** of the service request to decommission.	CreateTask ServiceOrderDecommission.xml

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

Configuration Audit

A configuration audit is executed as part of a service request deployment. During a configuration audit, ISC verifies that all Cisco IOS commands are present and that they have the correct syntax. A configuration audit also verifies that there were no errors during service deployment.

Table 3-9 Configuration Audit Task

className	Required Parameters	XML Examples
ServiceRequestDetails	• SubType=CONFIG_AUDIT • LocatorId **Note** The **LocatorId** of the service request used to deploy the configlets.	CreateTask ServiceOrderConfigAudit.xml

Service Deployment

A service deployment downloads the ISC-generated configlet, created for a service order, to the device configuration file.

You can specify the following options for service deployment:

- **ForceDeploy**—Takes a configlet for a service request that is already in the *Deployed* state and downloads it to the network device. Use **ForceDeploy** when a device configuration is lost or when you replace or change equipment.
- **IPSEC_Key_Regenerate**—Generates a new key for each tunnel in the service request and deploys the new key into the network.

Table 3-10 Deployment Task

className	Required Parameters	XML Examples
ServiceRequestDetails	• SubType=DEPLOYMENT • Provision • LocatorId **Note** The **LocatorId** of the service request used to deploy the configlets.	Create TaskServiceOrderDeployment.xml

IPsec Functional Audit

An IPsec functional audit is used to check the status of VPN tunnels for service requests that have already been deployed. The IPsec functional audit pings all nodes in the VPN to check connectivity and to ensure that the tunnels are up.

Note IPsec functional audits can be performed for IPsec site-to-site and IPsec-to-MPLS service requests only.

Table 3-11 IPsec Functional Audit Task

className	Required Parameters	XML Example
ServiceRequestDetails	• SubType= IPSEC_FUNCTIONALAUDIT • LocatorId **Note** The **LocatorId** of the service request used to deploy the configlets. • IPSecMirrorPing • VerifyAllInterfaces	CreateTask ServiceOrderIPsecFuncAudit.xml

MPLS Functional Audit

An MPLS functional audit verifies that the links in a service request or VPN are working correctly. The audit checks the routes to remote CPE devices in the VRF route tables on the PE devices.

Table 3-12 MPLS Functional Audit Task

className	Required Parameters	XML Example
ServiceRequestDetails	• SubType= MPLS_FUNCTIONALAUDIT • LocatorId **Note** The **LocatorId** of the service request used to deploy the configlets. • VPN • MPLSMirrorPing	CreateTask ServiceOrderMPLSFuncAudit.xml

SLA Collection

ISC uses the Cisco SA Agent MIB to monitor service level agreement (SLA) metrics. SLAs monitor the network performance by measuring response time, jitter, connect time, throughput, and packet loss. The SA Agent allows you to configure probes for performance measurements and uses a router monitor (RTRMON) MIB for access through SNMP. The MIB also supports SNMP notifications.

An SLA collection task collects all SLA data from SLA-enabled devices. ISC collects the relevant performance data and stores it persistently in the database. To view the data, you must execute a task to aggregate the data and run the SLA Report.

Note To collect SLA data from a device, it must have SA Agent and SNMP enabled, and SNMP parameters must already be set.

SLA collection includes performance data on Jitter (voice jitter) and the following protocols:

- Dynamic Host Configuration Protocol (DHCP)
- Domain Name System (DNS)
- File Transfer Protocol (FTP)
- Internet Control Message Protocol Echo (ICMP Echo)
- Transmission Control Protocol Connect (TCP Connect)
- User Datagram Protocol Echo (UDP Echo)
- Hypertext Transfer Protocol (HTTP)

Table 3-13 SLA Collection Task

className	Required Parameters	XML Example
ServiceRequestDetails	• SubType=SLA_COLLECTION • Device (or DeviceGroup) **Note** You must select at least one device or device group.	CreateTask ServiceOrderSLACollection.xml

General

The following general purpose XML API examples are provided with ISC:

- CreateConfigServiceOrderDownload—Creates a service order to download an inline configuration to a device.
- Create Inventory—Can be used to populate a database with sample inventory and is commonly used for testing or demonstration. The service order adds one type of each inventory element to the repository and enters the host name, domain name, management IP address, and the login and password for each device.
- CreateNPCInventory—Creates several physical links in a network, including the source and destination devices and relevant interfaces.
- CreateExecCommandServiceOrder—Creates a service order to execute a command on a device or device group. You can execute any command allowed by the device console.
- CreateServiceOrderResponse—Returns the following values for a service order:
  - **ServiceName** (for the service order)
  - **ServiceRequestName**
  - **LocatorId** (for the service request)
- CreateSLAInventory—Can be used to populate a database with sample inventory and is commonly used for testing. The service order adds the host name, domain name, and SLA data to devices in device groups.
- CreateResourceLock_Device, CreateResourceLock_Device_Batch, CreateResourceUnlock_Device, ViewResourceLock_Device—See the "Device Locking" section on page 5-21.
- DeleteServiceOrder—Deletes a service order.
- DeleteServiceOrder_purge—Deletes a service request of a specified subtype (**Type = QoS, L2VPN, Mpls, IPSec, IPSecRemoteAccess, Firewall, NAT**), with a given locator ID and **Force=true**.
- ViewConfiglet—View the ISC generated configlet for a specified service request. Returns the ViewConfigletResponse.
- ViewConfigletResponse—Returns the following values:
  - **ConfigletClob**
  - **LocatorId**
  - **Device**
  - **ConfigletText**—Contains the configlet.
- ViewConfiguration—View the configuration of a specified device. Returns the ViewConfigurationResponse.
- ViewConfigurationResponse—Returns the following values:
  - **LocatorId**
  - **Device**
  - **DeviceConfig**—Contains the device configuration.
- ViewCpe_propList_level—Provides a sample of a properties list and level usage. Returns the ViewCpe_propList_level_Response.

- ViewCpe_propList_level_Response—Returns the following values:
  - **LocatorId**
  - **Site**
  - **Device**
- ViewMPLSServiceRequest_propList_level—Provides a sample of a properties list and level usage. Returns the ViewMPLSServiceRequest_propList_level_Response.
- ViewMPLSServiceRequest_propList_level_Response—Returns the following values:
  - **OpType** (operation type; ADD or DELETE)
  - **LocatorId**
  - **MplsVpnLink** (and link attributes)
  - **CERCMembership**
- ViewServiceOrder—View the status of any service order, with a given service order name or locator ID. If the service order employs a task, the task locator ID is returned. The **TaskLocatorId** is required to view a task service order (**ViewTask**).