Firewall Provisioning

This feature is not supported in this release.

Cisco IP Solution Center (ISC) supports firewall services that include traffic filtering, inspection rules, and URL filtering. To provision firewall services using the API, you define the firewall policy and apply it to one or more Cisco firewall devices using a service request.

Use ISC to configure firewall rules on Cisco IOS devices running Cisco IOS Software version 12.2(13)T or later, and Cisco PIX Firewalls running software version 6.2 or later. ISC supports both stateful firewalls, such as context-based access control (CBAC), and stateless packet filtering firewalls, such as access control lists (ACLs).

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

For more information on firewall provisioning using ISC, refer to the *Cisco IP Solution Center Integrated VPN Management Suite Security User Guide, 4.0*.

This chapter contains the following sections:

Firewall Service Definitions

A firewall service definition (policy) defines access, inspection, and filtering rules, specifies the syslog server, messages, and log levels, and defines the authentication proxy server. Firewall service definitions can be used as parent or child policies for other service definitions.

- Access Control Lists (ACLs)—Filter network traffic by controlling whether IP packets are forwarded or blocked at a specific interface.
- Inspection Rules—Examine the protocol type and session information in outgoing packets to see if it matches certain criteria. If it does, return traffic of the same type is permitted into the network if it is associated with a session started within the firewall. ISC supports two types of inspection rules, CBAC for Cisco IOS devices, and Fixup for PIX Firewall devices.
- URL filtering—Allows you to prevent access to specific web sites.
- Syslog settings—Allows you to gather information about traffic and performance, analyze logs, and troubleshoot problems.

- Authentication Proxy—Allows you to apply specific security policies on a per-user basis instead of using a general policy.

Firewall Service Requests

You apply a firewall policy to one or more firewall devices using a service request. A firewall service request defines the firewall policy and the CPE devices to receive the policy. The CPE devices and template information are defined in the service request details using **FirewallLink**.

See the following example:

```
<objectPath xsi:type="ns1:CIMObjectPath">
              <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">ServiceDefinition</name>
                  <value xsi:type="xsd:string">TestFirewallPolicy</value>
                  <qualifier xsi:type="xsd:string">
                    <name xsi:type="xsd:string">ServiceDefinitionType</name>
                    <value xsi:type="xsd:string">Firewall</value>
                  </qualifier>
                </item>
              </properties>
              <objectPath xsi:type="ns1:CIMObjectPath">
              <className xsi:type="xsd:string">FirewallLink</className>
                <properties xsi:type="ns1:CIMPropertyList"
                        soapenc:arrayType="ns1:CIMProperty[]">
                    <item xsi:type="ns1:CIMProperty">
                      <name xsi:type="xsd:string">Cpe</name>
                      <value xsi:type="xsd:string">ensw2950-2</value>
                    </item>
                </properties>
                <objectPath xsi:type="ns1:CIMObjectPath">
                 <className xsi:type="xsd:string">LinkTemplate</className>
                 <properties xsi:type="ns1:CIMPropertyList"
                                          soapenc:arrayType="ns1:CIMProperty[]">
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">LogicalDevice</name>
                    <value xsi:type="xsd:string">1</value>
                  </item>
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">DatafilePath</name>
                    <value xsi:type="xsd:string">/nbi/AccessList</value>
                  </item>
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">DatafileName</name>
                    <value xsi:type="xsd:string">MyTemplate2</value>
                  </item>
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">TemplateActive</name>
                    <value xsi:type="xsd:string">true</value>
                  </item>
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">TemplateAction</name>
                    <value xsi:type="xsd:string">APPEND</value>
                  </item>
                </properties>
              </objectPath>
```

Provisioning Example

This section describes the process for using the API to provision firewall services, and includes the operation, object definition (className), and parameter definitions.

Process Summary

This firewall provisioning example uses the following processes:

- Prepare ISC inventory, including creating network objects and AAA servers.
- Create the firewall policy (service definition)
- Create the firewall service request (implemented as part of a service order)

Provisioning Process

This section describes the process for provisioning firewalls using XML examples.

The complete list of XML examples for firewalls are located at:
http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/isc/4_0/api/apiref/examples/index.htm

Note For clarity, this provisioning process shows each step as a separate XML request. Many of these steps can be combined using **performBatchOperations**.

Prepare Inventory

Use these steps to prepare the ISC repository for firewall provisioning.

Step 1 Create Devices.

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.

ISC supports these devices for firewall provisioning:

- Cisco IOS devices (**CiscoRouter**)
- PIX security appliances (**PIX**)

Table 12-1 Create Devices

Operation	className	Required Parameters
createInstance	CiscoRouter PIX	One or more of the following: • ManagementIPAddress • HostName • DomainName • Interface
	Interface	• Name • IPAddress • InterfaceEncapType (for **CiscoRouter**) • InterfaceType (for **PIX**)

XML Examples:

- CreateCiscoRouter.xml
- CreatePIX.xml

Step 2 Create customers (organizations).

A customer is a requestor of VPN services. Each customer can contain multiple customer sites. Each site belongs to only one customer and can contain multiple CPEs.

Table 12-2 Create Organization

Operation	className	Required Parameters
createInstance	Organization	• Name

XML Examples:

- CreateOrganization.xml

Step 3 Create sites and assign organizations to them.

Table 12-3 Create Sites

Operation	className	Required Parameters
createInstance	Site	• Name • Organization

XML Examples:

- CreateSite.xml

Step 4 Declare devices as CPEs and mark the interfaces. For firewall provisioning, you must define the firewall role (**FWRole**). Firewalls use outside, inside, and dmz interface names, or user-defined interface names.

- **outside**—interfaces on which VPN tunnels terminate
- **inside**—interfaces behind which the customer subnets reside
- **dmz** (demilitarized zone)—generally used for interfaces that separate areas within a corporate network

Table 12-4 Create CPE Devices

Operation	className	Required Parameters
createInstance	Cpe	• Site • Device • ManagementType
	Interface	• Name • IPAddressType (for **PIX**) • FWRole – inside – outside – dmz<number>

XML Examples:

- CreateCpe.xml

Step 5 Create a Network Object. This step is optional.

Network objects can be used in place of the **ServerAddress** parameter when defining source or destination devices in an access rules (**FWFilterRule**), or for URL filtering (**FWUrlServer**).

Table 12-5 Create Network Object

Operation	className	Required Keywords
createInstance	NetworkObject	• Name • Cpe • Type=HOST **Note** Use **HOST** to enter a specific IP address. Use **NETWORK** to enter a range of IP addresses. • Value

See the following example:

```
<ns1:createInstance>
    <objectPath xsi:type="ns1:CIMObjectPath">
     <className xsi:type="xsd:string">NetworkObject</className>
```

```
<properties xsi:type="ns1:CIMPropertyList"
              soapenc:arrayType="ns1:CIMProperty[]">
<item xsi:type="ns1:CIMProperty">
 <name xsi:type="xsd:string">Name</name>
 <value xsi:type="xsd:string">address_1</value>
 </item>
<item xsi:type="ns1:CIMProperty">
 <name xsi:type="xsd:string">Cpe</name>
 <value xsi:type="xsd:string">enqosce52</value>
</item>
<item xsi:type="ns1:CIMProperty">
 <name xsi:type="xsd:string">Type</name>
 <value xsi:type="xsd:string">HOST</value>
</item>
<item xsi:type="ns1:CIMProperty">
 <name xsi:type="xsd:string">Value</name>
 <value xsi:type="xsd:string">12.12.12.0</value>
</item>
 </properties>
 </objectPath>
</ns1:createInstance>
```

XML Example:

- CreateNetworkObject.xml

Step 6 Create the AAA server.

This step is only required if you use an authentication proxy server in your firewall policy. User profiles or group attributes can be obtained from the AAA server instead of being stored on the CPE device.

Table 12-6 Create the AAA Server

Operation	className	Required Parameters
createInstance	AAAServer	• Name • Organization • NumberOfRetries • Timeout • Address • AuthServerType= – RADIUS – NTDOMAIN – SDI – TACACS+ • Role= – AUTHENTICATION – ACCOUNTING – BOTH

XML Examples:

- CreateAAServer.xml
- CreateAAServerNTDOMAIN.xml
- CreateAAServerRADIUS.xml
- CreateAAServerSDI.xml
- CreateAAServerTACACS.xml

Creating a Firewall Service Definition

A firewall policy, or service definition, consists of access rules, inspection rules, URL filtering, syslog settings, and authentication proxy server information. See the following example:

```
<ns1:createInstance>
           <objectPath xsi:type="ns1:CIMObjectPath">
            <className xsi:type="xsd:string">ServiceDefinitionDetails</className>
            <properties>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">ParentPolicy</name>
                <value xsi:type="xsd:string">2</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">PermitIpsec</name>
                <value xsi:type="xsd:string">true</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">Organization</name>
                <value xsi:type="xsd:string">NbiCustomer</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">UrlFilterOn</name>
                <value xsi:type="xsd:string">true</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">SyslogOn</name>
                <value xsi:type="xsd:string">true</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">AuthProxyOn</name>
                <value xsi:type="xsd:string">true</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                <name xsi:type="xsd:string">TransparentModeOn</name>
                <value xsi:type="xsd:string">true</value>
              </item>
            </properties>
            <objectPath xsi:type="ns1:CIMObjectPath">
                <className xsi:type="xsd:string">FirewallFilterRule</className>
                <properties xsi:type="ns1:CIMPropertyList"
                            soapenc:arrayType="ns1:CIMProperty[]">
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">Source</name>
                    <value xsi:type="xsd:string">ANY</value>
                  </item>
                  <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">Destination</name>
                    <value xsi:type="xsd:string">ANY</value>
                  </item>
```

```
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Destination</name>
      <value xsi:type="xsd:string">192.168.200.41/24</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Destination</name>
      <value xsi:type="xsd:string">192.168.100.11/24</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">AccessDirection</name>
      <value xsi:type="xsd:string">inbound</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Protocol</name>
      <value xsi:type="xsd:string">AOL</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Protocol</name>
      <value xsi:type="xsd:string">BGP</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">ProtocolBundle</name>
      <value xsi:type="xsd:string">IPsecTraffic</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">ServiceDirection</name>
      <value xsi:type="xsd:string">normal</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Action</name>
      <value xsi:type="xsd:string">permit</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">InterfaceName</name>
      <value xsi:type="xsd:string">dmz2</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Overridable</name>
      <value xsi:type="xsd:string">true</value>
    </item>
  </properties>
</objectPath>
 <objectPath xsi:type="ns1:CIMObjectPath">
  <className xsi:type="xsd:string">FirewallInspectRule</className>
  <properties xsi:type="ns1:CIMPropertyList"
              soapenc:arrayType="ns1:CIMProperty[]">
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Application</name>
      <value xsi:type="xsd:string">http</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Port</name>
      <value xsi:type="xsd:string">111</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">PortEnd</name>
      <value xsi:type="xsd:string">156</value>
    </item>
 </properties>
</objectPath>
```

```
<objectPath xsi:type="ns1:CIMObjectPath">
  <className xsi:type="xsd:string">FWUrlServer</className>
  <properties xsi:type="ns1:CIMPropertyList"
              soapenc:arrayType="ns1:CIMProperty[]">
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">VendorType</name>
      <value xsi:type="xsd:string">websense</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Timeout</name>
      <value xsi:type="xsd:string">12000</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Interface</name>
      <value xsi:type="xsd:string">dmz1</value>
    </item>
  </properties>
  <objectPath xsi:type="ns1:CIMObjectPath">
  <className xsi:type="xsd:string">ServerDetails</className>
  <properties xsi:type="ns1:CIMPropertyList"
            soapenc:arrayType="ns1:CIMProperty[]">
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">ServerAddress</name>
      <value xsi:type="xsd:string">192.168.115.179</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Port</name>
      <value xsi:type="xsd:string">8002</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">ProtocolType</name>
      <value xsi:type="xsd:string">TCP</value>
    </item>
  <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">Port</name>
      <value xsi:type="xsd:string">8010</value>
    </item>
    <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">ProtocolType</name>
      <value xsi:type="xsd:string">UDP</value>
    </item>
  </properties>
  </objectPath>
  <objectPath xsi:type="ns1:CIMObjectPath">
    <className xsi:type="xsd:string">FWUrlExclusiveDomain</className>
    <properties xsi:type="ns1:CIMPropertyList"
                soapenc:arrayType="ns1:CIMProperty[]">
      <item xsi:type="ns1:CIMProperty">
        <name xsi:type="xsd:string">URL</name>
        <value xsi:type="xsd:string">smith@cisco.com</value>
      </item>
      <item xsi:type="ns1:CIMProperty">
        <name xsi:type="xsd:string">URLComment</name>
        <value xsi:type="xsd:string">Imaginary URL.</value>
      </item>
      <item xsi:type="ns1:CIMProperty">
        <name xsi:type="xsd:string">Action</name>
        <value xsi:type="xsd:string">permit</value>
      </item>
    </properties>
  </objectPath>
</objectPath>
```

```
      <objectPath xsi:type="ns1:CIMObjectPath">
        <className xsi:type="xsd:string">FWSyslog</className>
        <properties xsi:type="ns1:CIMPropertyList"
                    soapenc:arrayType="ns1:CIMProperty[]">
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">Facility</name>
            <value xsi:type="xsd:string">local0</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">WarningLevel</name>
            <value xsi:type="xsd:string">informational</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">TimeStamp</name>
            <value xsi:type="xsd:string">true</value>
          </item>
        </properties>
        <objectPath xsi:type="ns1:CIMObjectPath">
          <className xsi:type="xsd:string">FWLogServer</className>
          <properties xsi:type="ns1:CIMPropertyList"
                      soapenc:arrayType="ns1:CIMProperty[]">
            <item xsi:type="ns1:CIMProperty">
              <name xsi:type="xsd:string">Address</name>
              <value xsi:type="xsd:string">192.168.116.179</value>
            </item>
            <item xsi:type="ns1:CIMProperty">
              <name xsi:type="xsd:string">InterfaceName</name>
              <value xsi:type="xsd:string">outside</value>
            </item>
        </properties>
        </objectPath>
      </objectPath>
      <objectPath xsi:type="ns1:CIMObjectPath">
        <className xsi:type="xsd:string">FWAuthProxy</className>
        <properties xsi:type="ns1:CIMPropertyList"
                    soapenc:arrayType="ns1:CIMProperty[]">
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">AAAServer</name>
            <value xsi:type="xsd:string">1</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">LocalOrder</name>
            <value xsi:type="xsd:string">before</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">InterfaceType</name>
            <value xsi:type="xsd:string">inside</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">AuthProtocolList</name>
            <value xsi:type="xsd:string">http</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">AuthProtocolList</name>
            <value xsi:type="xsd:string">ftp</value>
          </item>
          <item xsi:type="ns1:CIMProperty">
            <name xsi:type="xsd:string">AuthProtocolList</name>
            <value xsi:type="xsd:string">telnet</value>
          </item>
        </properties>
        </objectPath>
      </objectPath>
    </objectPath>
```

Table 12-7 Create a Firewall Service Definition

Operation	className	Required Parameters
createInstance	ServiceDefinition	• Name • Type=Firewall • Global or Organization **Note** If you do not specify an **Organization**, the service policy is global. • ServiceDefinitionDetails
	ServiceDefinitionDetails	• ParentPolicy (if applicable) **Note** If policies conflict, the parent policies override the child policies. • PermitIpsec (**PIX** only) • CBAC (**CiscoRouter** only) • FirewallFilterRule **Note** Use **Overridable=true** to enable an access rule override the parent policy. • FirewallInspectRule • FWURLServer – ServerDetails – FWUrlExclusiveDomain • FWSyslog – FWLogServer • FWAuthProxy (**CiscoRouter** only)

Note To inherit attributes from parent policies, define a **ParentPolicy** in the service definition.

XML Examples:

- CreateFWServiceDefnAll.xml
- CreateFWServiceDefnSimple.xml

Creating a Firewall Service Request

A firewall service request consists of the firewall service policy, the firewall link, which defines the CPE device to be used as a firewall, and any template information.

Table 12-8 Create a Firewall Service Request

Operation	className	Required Parameters
performBatchOperations		
createInstance	ServiceOrder	• ServiceName • Organization • NumberOfRequests • ServiceRequest
	ServiceRequest	• RequestName • Type=Firewall • ServiceRequestDetails
	ServiceRequestDetails	• ServiceDefinition<choose a firewall policy> – ServiceDefinitionType=Firewall • FirewallLink – Cpe – LinkTemplate (optional) **Note** See the "Templates in a Service Request" section on page 4-9.

XML Examples:

- CreateFWServiceOrder.xml
- CreateFWServiceOrderwTemplate.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, an IPsec functional audit, or a certificate enrollment audit, see the "Tasks" section on page 3-8.