Cisco Security Manager 4.15 API Specification

(Version 2.4)

This document describes CSM message exchanges, XML schema, and client/server behavioral specifications in the Cisco Security Manager 4.15 Northbound API

Version 2.4: **Republished on May 10, 2017**

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Table of Contents

List of Figures

List of Tables

1 Overview

This document provides the protocol description and specification for the Cisco Security Manager (CSM) API as well as the behavioral requirements for any CSM client products that will use the API on the CSM Server and CSM Server (infrastructure device) itself. In addition, it provides the XML schema, which is the basis for the message content carried by the CSM northbound (NB) API.

The CSM NB API is designed to be used by client products that want to read network security configuration information or events, or that want to publish changes to those configurations.

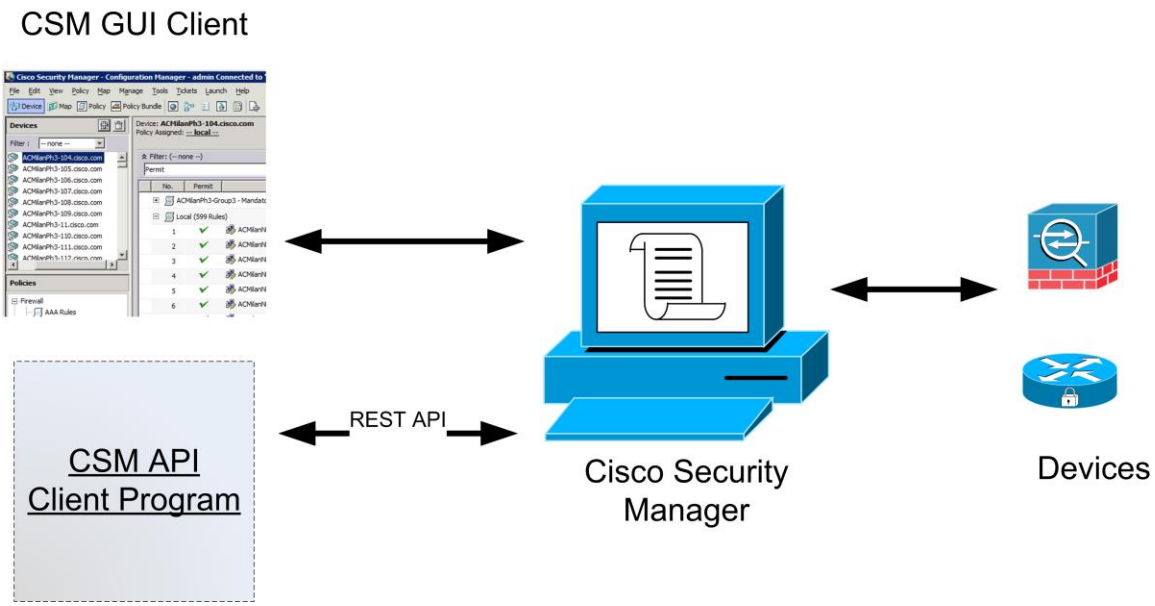
The CSM NB API is broken into services that enable various features to provide access to both global and device-specific network configuration policies for Cisco ASA and IPS devices.

After a client receives the policy information, it can use the information to perform security analysis, link to security events, and take other actions.

1.1 Scope

This document serves as the CSM NB API 2.1 specification--the specification for the XML schema which provides the basis of the message content carried in the CSM API payload and the behavioral specification for CSM client products and infrastructure products implementing the CSM NB API.

Note: Unless otherwise stated, all references to the CSM Client or the CSM "API" Client throughout this document refer to the third-party "CSM API Client Program" that uses the REST interface to communicate with the CSM Server. The CSM Client mentioned in this document should not be confused with the pre-bundled CSM GUI client application that is installed on user desktops to access the server. Wherever necessary, any references to the pre-bundled CSM GUI client application in the document includes a screen shot of the relevant screen to prevent any confusion. Please see the figure below where the interaction of the CSM API Client Program is clearly denoted.



1.2 Changes in Revision 1.1

The NB API 1.1 is a successor of the NB API 1.0. The following changes were added in the 1.1 version of the API:

1.2.1 Unified Access Rules

This is a new policy, supported in Version 1.1.

1.2.2 Security Policy Object

This is a new policy object, supported in Version 1.1.

1.2.3 Network object

This object has been changed to represent its data elements as <ipData>, in place of <ipv4Data>, when the API is used for Unified Access Rules.

1.2.4 Return user/ticket that last modified a config rule

One of the use cases of the API is to get configuration data from CSM for compliance checks. The configuration service API now also returns information regarding users and tickets that modified a configuration rule.

1.2.5 Add device status – up/down as part of the event service

Some Security Manager users have task threads that run daily and fetch changed configuration from the network devices managed by Security Manager. For this they use the execDeviceReadOnlyCLICmds API call to get the configuration from the Device.

The execDeviceReadOnlyCLICmds API call hangs if the Device Status is down. Hence, users interested in this area are advised to do a Device Status up/down check before executing the execDeviceReadOnlyCLICmds API call.

1.2.6 Exec command API call will be supporting custom timeouts

In the version of the API prior to 1.1 (which was Version 1.0), execDeviceReadOnlyCliCmds has no timeout value, so it can run for an infinite period of time and cause the API service to hang if a device is unresponsive. To prevent that problem, the subsequent version of the API (Version 1.1) allows the method execDeviceReadOnlyCliCmds to take an optional attribute in the Request. This optional attribute allows the API client to set an timeout for the execDeviceReadOnlyCliCmds method call.

1.2.7 API enhancement to return list of all the shared Policies defined in CSM

A new API was added in Version 1.1 that returns the list of all shared policies in the system for a given policy type.

1.2.8 Return the Device's SysObjectID in the Device Object

API Version 1.0 has many APIs which list out all the devices present in CSM. Additionally, API Version 1.1 now includes the SysObjectID in the response.

1.2.9 CSM Audit Logs should differentiate between logins through API and CSM client.

In CSM 4.3, logins to CSM from the NB API or a CSM Client would create an audit log message indicating a user login. However there was no differentiation/description in the audit log message indicating that a user had logged in using the NB API.

As part of the CSM 4.4 release, the audit log was enhanced to differentiate between logins through the API and logins through the CSM Client.

1.2.10 New Firewall Policies

The following new firewall policies are available beginning with CSM 4.4 and Version 1.1 of the API:

- InterfaceNAT64ManualFirewallPolicy. An InterfaceNAT64ManualFirewallPolicy represents a Unified (IPv6/IPv4) Manual NAT Rule.
- InterfaceNAT64ObjectFirewallPolicy. An InterfaceNAT64ObjectFirewallPolicy represents a Unified (IPv6/IPv4) Object NAT Rule.

1.3 Changes in Revision 2.0

The additions to this release of the NB API are described briefly in this section, and details of the individual APIs are covered in the later sections. The serviceVersion of the config service has been updated to 2.0.

1.3.1 Write API

In this release, write access for policies and objects is introduced. The following subsections provide more information:

1.3.1.1 Policy Objects

The support to add, modify, and delete policy objects is added in this release. This release also supports creating device overrides. The following policy objects are supported in this release :

1. Network
2. Service
3. Port-List
4. Interface Role
5. Time-range
6. Identity-User Group
7. Security-User Group

1.3.1.2 Policy

The support to add, delete, and modify a policy assigned to a device or a shared policy is added in this release. But the support to create/assign/inherit/unassign that shared policy via the API is not supported. The following policies are supported in this release:

1. Access-Rule
2. Unified Access-Rule
3. ACL Name Setting

1.3.1.3 Administration Page

To limit the payload size of the request, a new option has been added in the CSM API Administration page, where the user can specify a value between 1 and 5 MB, with the default value being 2 MB.

1.3.2 All CSM Server Mode Support

Beginning with Version 2.0, write API-related methods are introduced; they support the write APIs in all the CSM server modes, namely the following:

1. Ticketing disabled and workflow disabled
2. Ticketing enabled and workflow disabled
3. Workflow enabled and ticketing disabled
4. Workflow and Ticketing enabled

The CSMSession concept is introduced in the NB API. To use any of the Write API-related methods, a CSMSession needs to be created or opened from the API layer, and the unique ID that is returned should be used along with the write API methods. Internally, the createCSMSession API will create an activity or ticket based on the CSM server mode. Creating a CSMSession should be used even with non-workflow and non-ticketing mode. For more details about this, please see Section 8.1.3, Using CSMSession.

1.3.3 Deployment API

After the changes are submitted, immediate deployment for the affected devices can be initiated using the deployment APIs.

1.3.4 API to Read Policy Object

In this release, an API has been added to read the contents of a policy object given its name and type, or given only its ID.

1.3.5 Access-Rule Changes

Beginning with this release, the read output of the access rule is modified to include the policy name and the section information. When a device is inheriting a shared policy, this addition will differentiate the local rules from the shared policy rule. Two new elements, <sectionName> and <policyName>, have been added. The <sectionName> tags give the section detail to which the rule belongs, and <policyName> gives not only the name of the shared policy but also the complete path so that it helps in understanding the inheritance model. The following example shows the newly added element:

```
<deviceAccessRuleUnifiedFirewallPolicy>
   <gid>00000000-0000-0000-0000-012884901983</gid>
   <name/>
   <lastUpdateTime>2014-03-03T13:00:28.57Z</lastUpdateTime>
   <updatedByUser>asbabu</updatedByUser>
   <lastCommitTime>2014-03-03T13:00:37.142Z</lastCommitTime>
   <ticketId>asbabu_03.Mar.2014_18.30.15</ticketId>
   <type>DeviceAccessRuleUnifiedFirewallPolicy</type>
   <orderId>0</orderId>
   <isMandatoryAggregation>true</isMandatoryAggregation>
   <description> </description>
   <configState>committed</configState>
   <sectionName>london</sectionName>
   <policyName>Global/Europe/UK</policyName>
   <isEnabled>true</isEnabled>
   <direction>In</direction>
   <permit>false</permit>
   <interfaceRoleObjectGIDs>
   <gid>00000000-0000-0000-0000-000000000213</gid>
   </interfaceRoleObjectGIDs>
   <sources>
   <networkObjectGIDs>
   <gid>00000000-0000-0000-0000-000000000102</gid>
   </networkObjectGIDs>
   </sources>
   <destinations>
   <networkObjectGIDs>
   <gid>00000000-0000-0000-0000-000000000102</gid>
   </networkObjectGIDs>
   </destinations>
   <services>
   <serviceObjectGIDs>
   <gid>00000000-0000-0000-0000-000000001041</gid>
   </serviceObjectGIDs>
   </services>
   <logOptions>
   <isFirewallLoggingEnabled>true</isFirewallLoggingEnabled>
   <isDefaultLogging>true</isDefaultLogging>
   </logOptions>
</deviceAccessRuleUnifiedFirewallPolicy>
```

Figure 1: Access-Rule Changes in Version 2.0

1.4 Changes in Revision 2.1

In this revision, the following CSM Configuration Service API methods for working with shared policies are introduced.

1.4.1 CreateSharedPolicy

The CreateSharedPolicy method is used to create a new shared policy.

1.4.2 DeleteSharedPolicy

The DeleteSharedPolicy method is used to delete an existing shared policy.

1.4.3 RenameSharedPolicy

The RenameSharedPolicy method is used to rename an existing shared policy.

1.4.4 AssignSharedPolicy

The AssignSharedPolicy method is used to assign a shared policy to devices. Assigning a shared policy will replace the existing local policy for the device.

1.4.5 UnassignSharedPolicy

The UnassignSharedPolicy method is used to unassign a shared policy.

1.4.6 InheritSharedPolicy

The InheritSharedPolicy method is used to inherit a shared policy from another shared policy. If no inheritance option is selected then the inherited policies will be detached.

1.5 Changes in Revision 2.2

In this revision, the following CSM Configuration Service read API method for getting list of policy objects for a specific object type is introduced.

1.5.1 getPolicyObjectsListByType

this API to get the list of policy objects based on type. User can readily use the list and use the object of interest for processing in other Config API.

1.6 Audience

This document is technical in nature and is intended for architects, development engineers, and technical marketing engineers.

1.7 References

The following are referenced within this document and form a normative part of this specification to the extent specified herein. In the event of a conflict with this specification and the following referenced specifications, the contents of this specification take precedence.

- RFC-3986, Uniform Resource Identifier (URI): Generic Syntax, Berners-Lee, Fielding and Masinter, January 2005
- RFC-3579, RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP), Aboba and Calhoun, September 2003
- RFC-2387, The MIME Multipart/Related Content-type, Levinson, August 1998
- XML-binary Optimized Packaging, Gudgin, Mendelsohn, Nottingham and Ruellan, http://www.w3.org/TR/xop10/, January 25, 2005
- Describing Media Content of Binary Data in XML, Karmarkar and Yalçınalp, http://www.w3.org/TR/xml-media-types/, May 4, 2005
- RFC-2616, Hypertext Transfer Protocol -- HTTP/1.1, Fielding, Gettys, Mogul, Frystyk, Masinter, Leach and Berners-Lee, June 1999
- XML Path Language http://www.w3.org/TR/xpath/, November 1999

1.8 Glossary

The following list describes acronyms and definitions for terms used throughout this document:

Abbreviation	Definition
AAA	authentication, authorization and accounting
API	Application Programming Interface
AS	authentication server (e.g. AAA)
AUS	Auto Update Server
CA	certificate authority. May also refer to "Configuration Archive" module in CSM that stores raw device configuration data.
CNS	Cisco Networking Services
CR	certificate repository
CSM	Cisco Security Manager
DNS	domain name service
FQDN	fully qualified domain name
HPM	Health and Performance Monitoring application of CSM
HTTP	Hyper Text Transfer Protocol
IP	Internet protocol
LAN	local area network
NAS	network authentication server
NB	North Bound
OOB	Out of Band (uncontrolled changes done outside of CSM directly on the device)
PKI	public key infrastructure
PKC	public key cryptography
SDK	Software Development Kit
UI	user interface
URI	universal resource identifier
XML	extensible markup language
XPath	XML Path Language
REST	REpresentational State Transfer

Table 1: Glossary Terms

1.9 Conventions

The following textual conventions are observed in this document:

- All type definitions start with an uppercase letter, as in the following example:
  - `xs:simpleType name="ObjectIdentifier"`
- All element names start with a lowercase letter, as in the following example:
  - `<xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>`

1.10 Overview of CSM Message Flows

When a CSM client wishes to use the CSM server, some of the main high-level message flows are shown in Figure 2. The CSM Server is configured to know about the devices in the network and reads those configurations for Firewalls, IPS devices, etc. After initiation, any CSM client may authenticate to the CSM server and access the methods provided in the API. The CSM interface is defined as multiple services such as common (Section 2), config (Section 3), events (Section 3.2.11) and utility (Section 5) services.

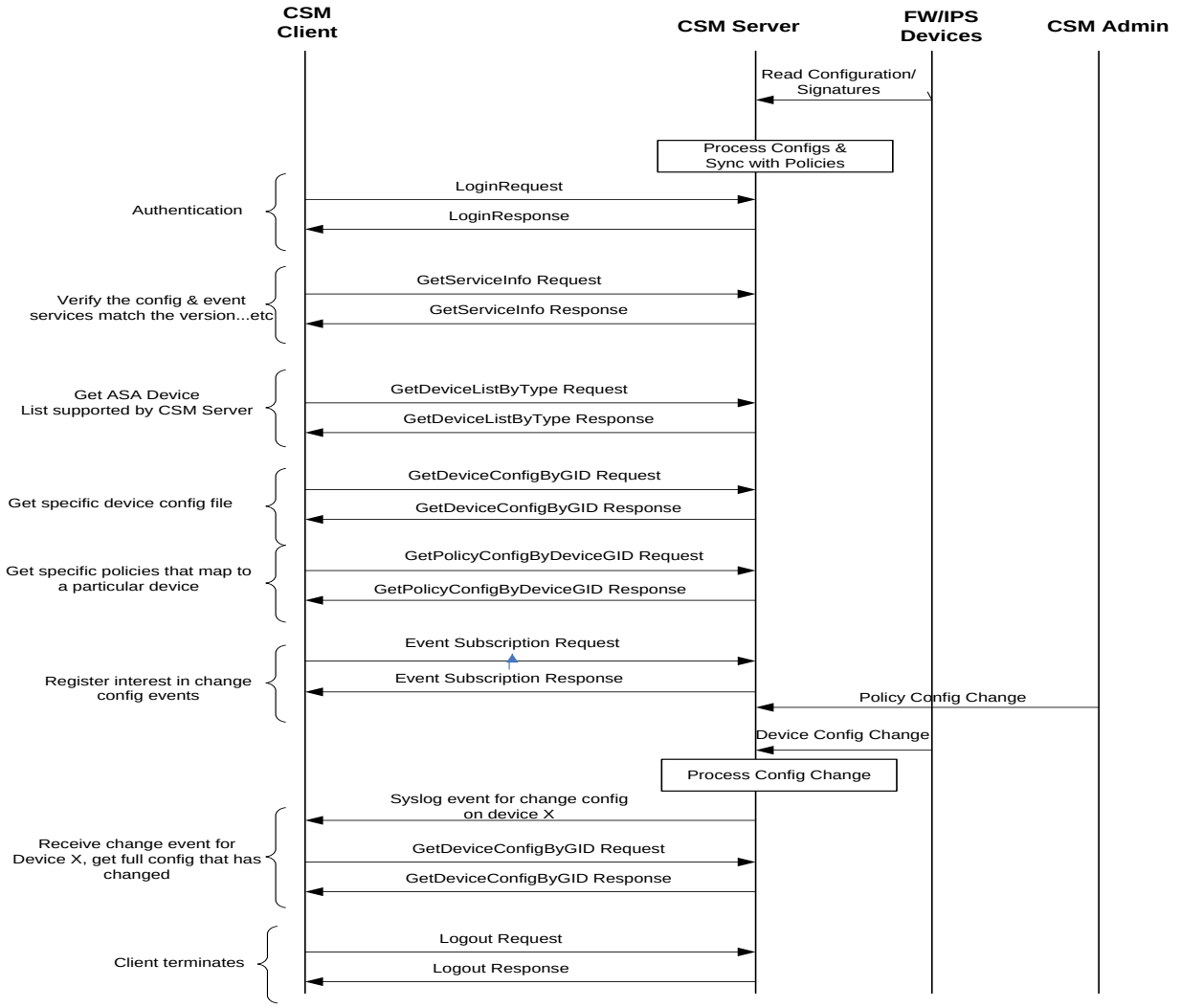


Figure 2: High-Level CSM Message Exchange

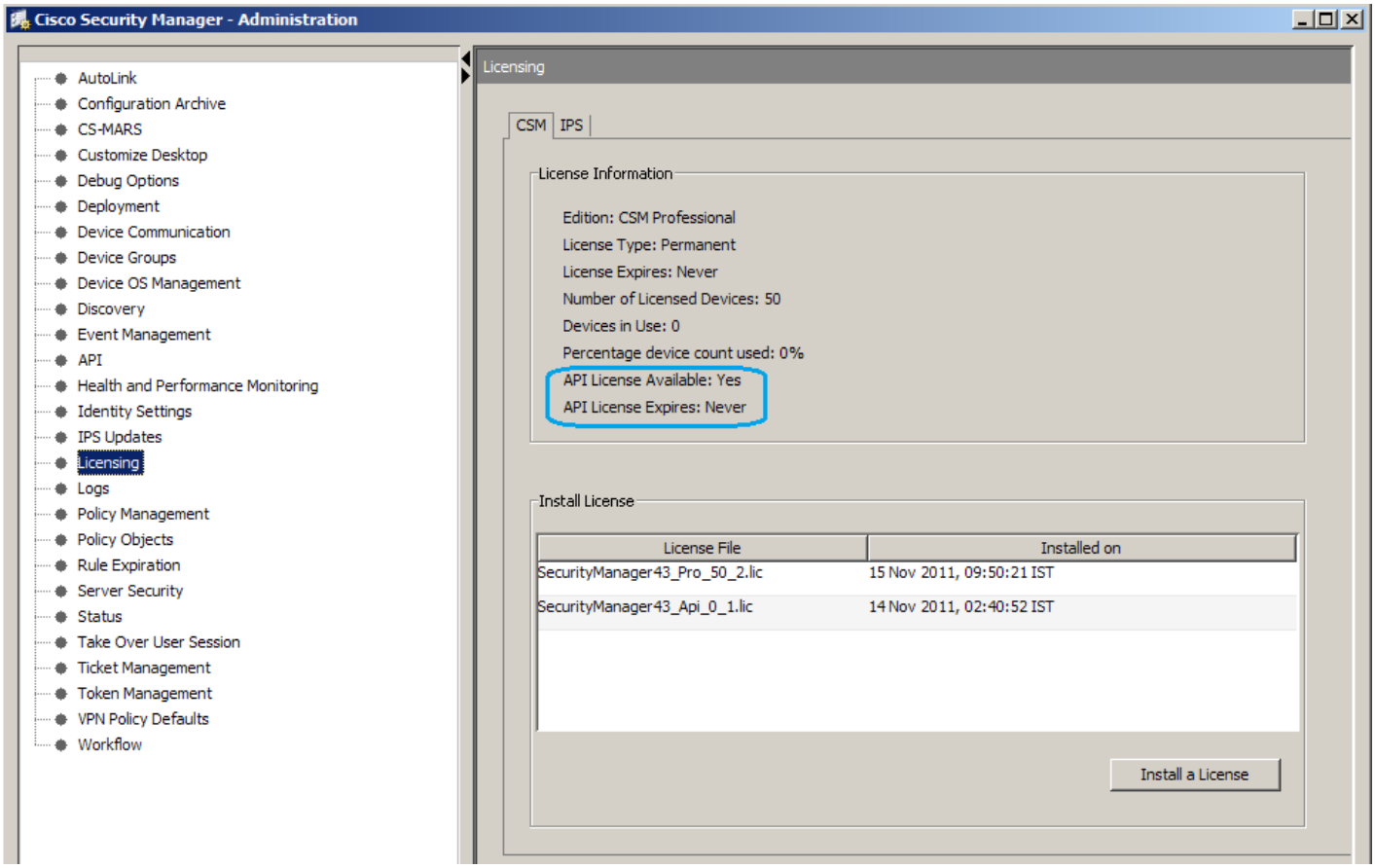
REST is used for the transport of CSM messages. In other words, a CSM message is a URL accessed by the CSM client. The XML schema for CSM messages is provided in each service section in this document as well as the complete XSD files in Section 11.

The CSM API is structured with four major subcategories:

- **Common Service API**: Contains common method definitions and schema details (related to login, logout, ping/heartbeat, etc.). For more details, see the section titled Common Service API.
- **Config Service API**: Contains methods and schema to retrieve and write device configuration data. For more details, see the section titled CSM Configuration Service API.
- **Event Service API**: Contains methods and schema that allow the client to subscribe for event notifications in specific cases (deployment and out-of-band change events, etc.). For more details, see the section titled CSM Events Service API.
- **Utility Service API**: Contains methods and schema that allow the user to execute specific CLI commands on the device using the CSM API interface. For more details, see the section titled CSM Utility Service API

1.11 Licensing

This feature is licensed. A specific CSM API license must be applied via the CSM **Tools → Security Manager Administration → Licensing** page. The API license can only be applied on a server licensed for the CSM professional edition. The license cannot be applied on CSM running a Standard edition of the license or when it is in evaluation mode.



All API requests without a valid license will return an error response containing:

Error code: 26

Error Description: API license is not enabled. Please add a valid API license and retry this operation.

Workaround: To resolve this issue, clear all cache from the browser and restart CSM.

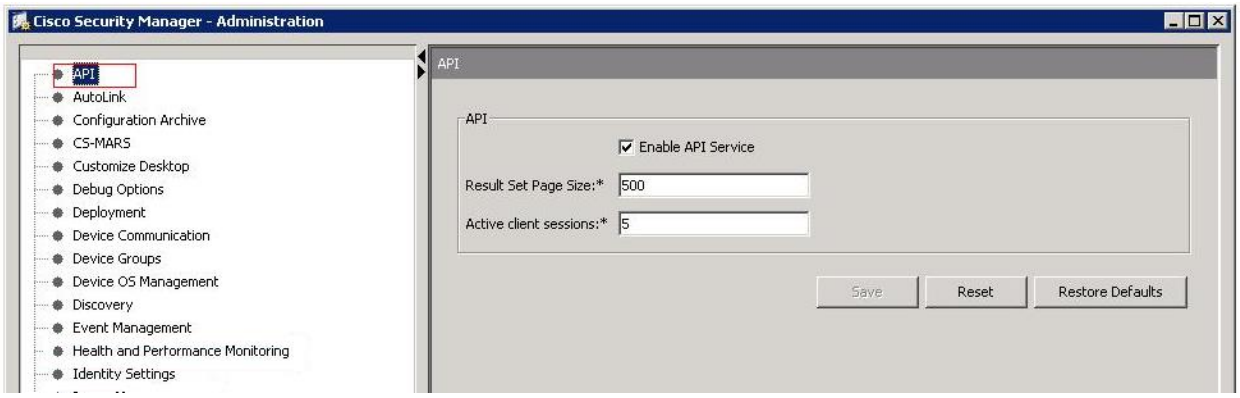
1.12 Prerequisites

Following are some prerequisites to use and work with the API:

- The latest version of the CSM Server that includes the API support must be installed. Please refer to the CSM installation guide for more details.
- After the server is installed, managed devices (e.g., ASA and IPS) must be added to CSM before the corresponding policy data is queried using the API.
- API will only return data that is "committed" to the policy database. For that reason, applicable activities must be submitted before the data is visible via the API.
- Health and Performance Monitor (HPM) should be enabled for out-of-band (OOB) and device status UP/DOWN change notifications to work.
- API does not include any SDK for any specific programming language. The API exposes a REST (Representational State Transfer) -based interface, so API clients can be implemented in any language as long as the XML Message protocol is adhered to.
- All API Services are accessible only using HTTPS. No HTTP access is allowed. The "root" URL for REST based API access is *https://<server-ip or host>/nbi/.*

1.13 API Administration Settings

The following global administration settings are specific to the API feature (Please refer to the version-specific CSM user guide and installation guide for additional information):

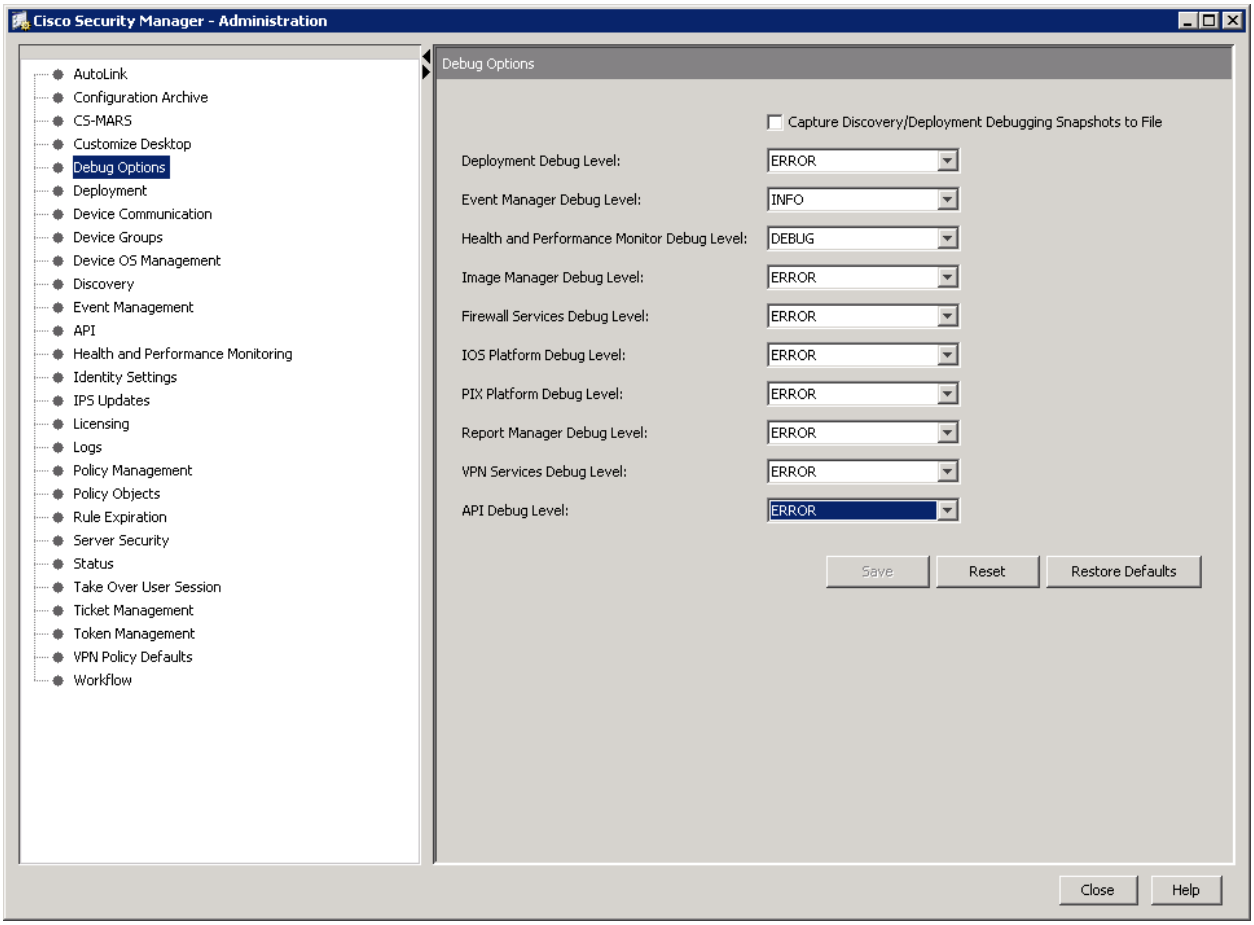


- **Enable API Service**: Allows the API feature to be completely enabled/disabled
- **Result Set Page Size**: Controls the size of the result set that is returned in a single response. Values are 100 (minimum), 500 (default) and 1000 (maximum). This configuration applies only to the **GetPolicyConfigurationByName** (Section 3.2.8) and **GetPolicyConfigurationByDeviceGID** (Section 3.2.9) methods. Also, see Section 2.2.1.1 for details on how responses are paginated.

- **Active Client Sessions**: Controls the total number of simultaneous active API client sessions (logins) that are allowed. Values are 1 (minimum), 5 (default), and 10 (maximum).

1.14 Debug Settings

If you want to enable debugging on the API requests on the CSM server, please set the value of the API Debug Level field to DEBUG. (The default value of this field is ERROR.)



2 Common Service API

This section describes the common methods and common object model for all services on the CSM API.

2.1 Object Model

The following object classes are used throughout the API specification.

2.1.1 Object Identifier

The Object Identifier is a global unique identifier for an object. The Object Identifier is a 128-bit value based on RFC4122.

```
<xs:simpleType name="ObjectIdentifier">
      <xs:restriction base="xs:string">
         <xs:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}"
         > </xs:pattern>
      </xs:restriction>
   </xs:simpleType>

   <xs:complexType name="ObjectIdentifierList">
            <xs:sequence>
                  <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
   </xs:complexType>
```

Figure 3: ObjectIdentifier & ObjectIdentifierList XML Schema

2.1.2 Base Object

The common class for all objects in the system and all object classes inherit from this class. It has the following attributes:

Attribute	Type	Comment
gid	Object Identifier	Unique object identifier for an object and is immutable for the object lifetime
lastUpdateTime	TimeStamp	Indicates the update time for an object either created or updated.
name	String	An optional display name for the object.
parentGID	Object Identifier	An optional parent object identifier that identifies an object's parent instance.
updatedByUser	String	Username of the User who updated the Object.
lastCommitTime	dateTime	Last Updated time for the object
ticketId	String	Ticket ID of the ticket as part of which the Object has been updated. This will not be available if Ticketing is not enabled.
activityName	String	Activity Name of the Activity as part of which the Object has been updated. This is applicable if the CSM is in WorkFlow mode.

Table 2: BaseObject Class Attributes

```
<xs:complexType name="BaseObject" >
      <xs:sequence>
            <xs:element name="gid" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastUpdateTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="parentGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
            <xs:element name="updatedByUser" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastCommitTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="ticketId" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="activityName" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
</xs:complexType>
```

Figure 4: BaseObject XML Schema

2.1.3 Device

A device object is used to represent a single device in the system. Virtual contexts are also modeled as individual devices in the system. The device object also expresses a relationship between a parent (admin/system context) and its associated child contexts.

The Device class inherits from BaseObject including all attributes.

Element/Attribute	Type	Comment
osType	String	The device OS Type from an enumerated list {IOS, FWSM, ASA, PIX, IPS}
osVersion	String	The software version of the OS running on the device. Version string could be 6.1, 6.2 etc. on a PIX platform, 12.1, 12.2S etc. on an IOS platform etc.
imageName	String	The OS Image name.
mgmtInterface	Interface	A reference to the management interface for the device. The interface which is used to manage the device in CSM is taken as management interface.
interfaceList	Sequence of Interface	A list of interfaces within the device. (*Contains list of interfaces other than management interface of the device.)*
fullConfig	String	An element containing the full configuration of the device. The full configuration of the device will be represented by an ASCII output of the show running config CLI command. This is only shown when the *getDeviceConfigByGid* or *getDeviceConfigByName* API's to get the full config of the device. In all other cases this element value is not set.
virtualContextList	List of Device	A list of virtual context objects belonging to the device
configState	ConfigurationSTate	The configuration state of the device and whether there are any uncommitted changes. Value taken from { committed, deployed

		}
sysObjectID	String	System Object ID of the Device.

Table 3: Device Class Attributes

```
<xs:complexType name="Device">
    <xs:complexContent>
        <xs:extension base="BaseObject">
            <xs:sequence>
                <xs:element name="osType" type="OSType" minOccurs="1" maxOccurs="1"/>
                <xs:element name="osVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="imageName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="sysObjectID" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="fullConfig" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="mgmtInterface" type="Interface" minOccurs="0" maxOccurs="1"/>
                <xs:element name="interfaceList" type="InterfaceList" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="virtualContextList" type="Device" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="configState" type="ConfigurationState" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
```

Figure 5: Device XML Schema

2.1.3.1 Interface

The Interface class defines a network interface in a network device. The InterfaceList class defines a sequence of Interface instances.

Attribute	Type	Comment
type	String	The type of network interface. Following are the supported interface type values – *Null, Management, Analysis-module, Async, ATM, BRI, BVI, Content-engine, Dialer, Dot11Radio, Ethernet, FastEthernet, GigabitEthernet, TenGigabitEthernet, HundredGigabitEthernet, FDDI, Group-Async, HSSI, IDS-Sensor, Loopback, Multilink, Port-channel, POS, PRI, Serial, Switch, Tokenring, Tunnel, VG-anylan, Virtual-Template, Virtual-TokenRing, VLAN, Redundant*
identifier	String	The identifier for the network interface. If name is configured for any interface on the device then Interface Identifier value will be taken as name else the interface type/port/slot will be shown as identifier. For example a GigabitEthernet interface is configured as *outside* then the API will show the identifier as *outside* and type as *GigabitEthernet*.
ipInterface.domainName	String	An optional DNS domain name.
ipInterface.ipAddress	String	The IP Address configured on the interface including its mask. The mask is not stored for IPS devices.
ipInterface.isNatAddress	String	True if the management IP address is a NATed address. (This placeholder element is not used in Version 1.1 of the API)
ipInterface.realIpAddress	String	In case the management IP address is a NATed address, the realIpAddress is the non NATed ip address of the management interface. (This placeholder element is not used in Version 1.1 of the API)
macInterface.macAddress	String	The mac address of the interface (optional). The MAC Address for ASA/PIX devices alone is stored. For others device types, CSM does not store the MAC address. Also CSM does not manage the burnt in MAC Address details. CSM manages only the mac - address that is changed explicitly by the user, so response will have only this data.

Table 4: Interface Class Attributes

```
<xs:complexType name="InterfaceList">
        <xs:sequence>
            <xs:element name="interface" type="Interface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Interface">
        <xs:sequence>
            <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="identifier" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="ipInterface" type="IPInterfaceAttrs" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="macInterface" type="MACInterfaceAttrs" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MACInterfaceAttrs">
        <xs:sequence>
            <xs:element name="macAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IPInterfaceAttrs">
        <xs:sequence>
            <xs:element name="domainName" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="ipAddress" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="isNatAddress" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="realIpAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

Figure 6: Interface XML Schema

2.1.3.2 Firewall Capabilities

Attribute	Type	Comment
fwOsMode	Int	Indicates if the FWSM/ASA is running in Transparent (1) or Routed (2) mode.
fwOsMultiplicity	Int	Returns if the FWSM/ASA is running in a single (1) or multi (2) context.
contextName	String	Returns the name of the context. If the current device represents a context.
isComposite	Boolean	Indicates if the device is a “composite” or not. Catalyst 6500 containing FWSM or other blades are marked as composite.

Table 5: FirewallCapabilities Class Attributes

2.1.4 DeviceGroup

A device group object is used to represent a container of devices in the system. Device groups contain zero or more devices and zero or more children device groups.

The DeviceGroup class inherits from BaseObject including all attributes.

Element/Attributes	Type	Comment
Element:path	String	A hierarchichal path string indicating the group.
Element: device	Device	Zero or more devices within this device group
Element: deviceGroup	DeviceGroup	Zero or more device groups within this device group

Table 6: DeviceGroup Class Attributes

```
<xs:complexType name="DeviceGroup">
        <xs:complexContent>
            <xs:extension base="BaseObject">
                <xs:sequence>
                    <xs:element name="path" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="device" type="Device" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="deviceGroup" type="DeviceGroup" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="DeviceGroupPath">
        <xs:sequence>
            <xs:element name="pathItem" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
```

Figure 7: DeviceGroup and DeviceGroupPath XML Schema

2.1.5 Port Identifier

The PortIdentifier class defines a physical or virtual port identifier in a network device.

Attribute	Type	Comment
slotNum	unsignedInt	The identifier of the slot in the case of a modular chassis. In a non-modular chassis this attribute is empty.
moduleNum	unsignedInt	The identifier of the module in the case of a sub-module within a slot. In a non-modular chassis this attribute is empty.
portNum	unsignedInt	The port number

Table 7: PortIdentifier Class Attributes

```
<xs:complexType name="PortIdentifier">
        <xs:sequence>
            <!-- for non-modular chassis or chassis with a continuous port numbering scheme slot/module are
not included -->
            <xs:element name="slotNum" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
            <xs:element name="moduleNum" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
            <xs:element name="portNum" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

Figure 8: PortIdentifier XML Schema

2.1.6 BaseError

The common class for all request errors is defined by the BaseError class. It has the following attributes:

Attribute	Type	Comment
Code	Unsigned Long	A unique error code that identifies the type of error that occurred with the request.
Description	String	A description of the error that occurred

Table 8: BaseError Class Attributes

```
<xs:complexType name="BaseError">
    <xs:sequence>
        <xs:element name="code" type="xs:unsignedLong" maxOccurs="1"></xs:element>
        <xs:element name="description" type="xs:string" maxOccurs="1"></xs:element>
    </xs:sequence>
</xs:complexType>
```

Figure 9: BaseError XML Schema

The following general error codes are currently defined:

Code	Description
0	Reserved
1	General Failure
2	Lack of resources
3	Object Creation Failure
4	Authorization Failure: No session found
5	Authorization Failure: Invalid or expired session
6	Internal communication failure
13	XML request payload contains no data.
15	XML request is invalid. (*This error is caused if the XML request does not adhere to the published XML schema or if the XML is illformed or invalid. Though this is a system wide error, this error is usually set inside a specific response object if the application is able to parse the request method that is being called*)
17	protVersion is optional, and if an unsupported version is specified, then this error is returned. In this release, the supported version is Version 1.0.
21	Internal Error
25	API Service is disabled
26	API license is not applied

Table 9: System Error Codes

Additional method specific errors codes are defined in the respective sections. All response objects extend from the Base Error object. The error content is set in cases the CSM API encounters any error when servicing a method.

The error messages returned are of two general types:

1. **Common/System wide errors** : As defined in the table above. These errors are generally due to some unrecoverable errors encountered by the application. These errors can occur when servicing any request.

2. **Method Specific errors** : These are application errors that are specific to method being processed.

The system wide errors return the Base Error content as the only response:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:baseError xmlns:ns1="csm">
        <code>1</code>
        <description>General Failure</description>
</ns1:baseError>
```

Method specific errors encode the Base Error content inside the response object. Following is an example of an error encountered when servicing a login request:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:loginResponse xmlns:ns1="csm">
        <protVersion>1.0</protVersion>
        <error>
                <code>7</code>
                <description> Authentication Failure: Invalid username and/or password specified.</description>
        </error>
        <serviceVersion>1.0</serviceVersion>
        <sessionTimeoutInMins>15</sessionTimeoutInMins>
</ns1:loginResponse>
```

2.2 Methods

2.2.1 Common Request & Response

All methods in this API take an XML object as an argument for the method request and response. The XML object passed as part of the request (and echoed in the response) are derived from the following class which includes the following attributes.

Attribute	Type	Comment
protVersion	double	identifies the version of the protocol associated with the particular request/response being sent
reqId	String	identifies a unique token sent by the client in the request that the server will echo in the associated response
startIndex	Unsigned Long	An optional start index that is specified by the “client” requests that are “paginated”. This is applicable for policies like firewall rules that might return a large number of rows of data. The client must set the startIndex equal to the end index of the previous response to fetch the next page of data.
endIndex	Unsigned Long	An optional end index that is specified by the “server” when it has not returned “all the data”.
totalCount	Unsigned Long	An optional total count from the “server” that indicates the total number of rows in this policy.
Error	BaseError	identifies any errors that may occur upon a request submitted to the server

Table 10: BaseReqResp Class Attributes

```
<xs:complexType name="BaseReqResp" >
    <xs:sequence>
        <xs:element name="protVersion" type="xs:double" minOccurs="0" maxOccurs="1"/>
        <xs:element name="reqId" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="startIndex" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
        <xs:element name="endIndex" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
        <xs:element name="totalCount" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
        <xs:element name="error" type="BaseError" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
```

Figure 10: BaseReqResp XML Schema

2.2.1.1 Pagination

Some of the service calls (say fetching a list of access rules) might return a large set of data. To prevent such large fetches from causing performance issues on the server and client, these results will be ‘paginated’. Pagination is

applicable only for the *GetPolicyConfigByName* and *GetPolicyConfigByDeviceGID* methods. The pagination scheme will work as follows:

1) In the first request the client sends, none of the parameters startIndex, endIndex will be set. This is an indication to the server that this is a fresh request.

2) If the server determines that this request needs to be paginated (because the total result set is greater than the page size) then it will return a paged result and will set the following two elements:

a. endIndex: Will be set to endIndex of the current result set. For example if this is the first request and 1000 rows are being returned then endIndex will be 1000.

b. totalCount: Will contain the total count of the query itself. For example, if the total result contains 10,000 rows, then the element totalCount will be set to 10,000.

3) In the subsequent request, the client must set the startIndex in the BaseReqResp object and send the same query request again. In this case the startIndex of the client request will be equal to the endIndex of the last response received.

4) In the final response if there are no more data to be fetched, the server **will not set** any data for endIndex and totalCount.

Consider an example where a client is querying Firewall rules on a device "A" which has 3600 rules. The page size configured in the system is 1000. Following are the sequence of calls:

- Client → Get Firewall Rules on device "A"
- Server → Response with 1000 rules and endIndex=1000 and totalCount=3600
- Client → Get Firewall Rule on device "A" and startIndex=1000
- Server → Response with 1000 rules and endIndex=2000 and totalCount=3600
- Client → Get Firewall Rule on device "A" and startIndex=2000
- Server → Response with 1000 rules and endIndex=3000 and totalCount=3600
- Client → Get Firewall Rule on device "A" and startIndex=3000
- Server → Response with 600 rules and endIndex=<not-set> and totalCount=<not-set>

As a general case, if the client notices that the endIndex or totalCount is not set then all rows are expected to have been returned.

2.2.2 Method login

The login method authenticates a CSM client attempting to access the services provided by the CSM server. This method must be called prior to any other method called on other services. Since the introduction of write APIs from Version 2.0, the following is the behavior of the login method in different CSM server modes and between CSM GUI client login and API login. In the below table the CSM GUI Client denotes the pre-bundles CSM thick client along with CSM installer and API denotes the client that used CSM API.

Scenario	Workflow	Ticketing	Behavior

Login via CSM GUI client and API from same user id	Enabled	Enabled	Allowed
Login via CSM GUI client and API from different user ID	Enabled	Enabled	Allowed
Login via API from parallel sessions with same user id	Enabled	Enabled	Allowed
Login via CSM GUI client and API from same user id	Enabled	Disabled	Allowed
Login via CSM GUI client and API from different user ID	Enabled	Disabled	Allowed
Login via API from parallel sessions with same user id	Enabled	Disabled	Allowed
Login via CSM GUI client and API from same user id	Disabled	Enabled	Not allowed
Login via CSM GUI client and API from different user ID	Disabled	Enabled	Allowed
Login via API from parallel sessions with same user id	Disabled	Enabled	Not allowed
Login via CSM GUI client from parallel session with same user id without API configuration	Disabled	Enabled	Allowed
Login via CSM GUI client and API from same user id	Disabled	Disabled	Not allowed
Login via CSM GUI client and API from different user ID	Disabled	Disabled	Allowed
Login via API from parallel sessions with same user id	Disabled	Disabled	Not allowed

2.2.2.1 Request

An example of the method login request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/login

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<csm:loginRequest xmlns:csm="csm">
<protVersion>1.0</protVersion>
<reqId>123</reqId>
<username>admin</username>
<password>admin</password>
<heartbeatRequested>true</heartbeatRequested>
<callbackUrl>https://10.106.134.111/nbi/login</callbackUrl>
</csm:loginRequest>
```

Figure 11: login Request Example

Table 11: login Request Elements and Attributes Descriptions

XML Argument	Definition
loginRequest	login request authenticates the client against the server and returns a cookie that should be used in subsequent requests
username	The CSM client username associated with the session
password	The CSM client password associated with the session
heartbeatRequested	This attribute may be optionally defined. If the attribute is set to true then the CSM client will receive a heartbeat callback from the CSM server. The server will try to ping the client with a frequency close to *(inactivity timeout) / 2* minutes. If the client does not respond to the heartbeat then the API retries the heartbeat during the next interval. If the heartbeat is successful then the session inactivity timeout is reset.
callbackUrl	The URL at which the CSM server will make the callback. This needs to be specified if the heartbeatRequested is true. Only HTTPS based callback URLs are allowed
HTTP Method	POST
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="loginRequest" type="LoginRequest"/>
<xs:complexType name="LoginRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence>
                        <xs:element name="username" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="password" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="heartbeatRequested" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
                        <xs:element name="callbackUrl" type="xs:string" minOccurs="0" maxOccurs="1"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
```

Figure 12: LoginRequest XML Schema

2.2.2.2 Response

The Login API validates the user credentials and returns a session token as a secure cookie. The session value is stored under the "asCookie" key. This session has a default session inactivity timeout of 15 minutes. An example of the HTTP Header and XML content response is shown in figure below. The fields in these messages are described in table below.

```
HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Content:

<?xml version="1.0" encoding="UTF-8"?>
<loginResponse>
      <protVersion>1.0</protVersion>
      <serviceVersion>1.0.1</serviceVersion>
      <sessionTimeoutInMins>15<sessionTimeoutInMins>
</loginResponse>
```

serviceVersion will be 2.0 for the config Service:

Figure 13: login Response Example

Table 12: login Response Elements and Attributes Description

HTTP/XML Response	Definition
HTTP Header: **asCookie**	A cookie defining the authentication string, expiration date, path and domain that must be passed in all subsequent method calls e.g. Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/; domain=.hostdomain.com
XML Element: loginResponse	XML content that returns the session information or failure
serviceVersion	The service version of the Configuration service that is running. This attribute is included only if the user is successfully authenticated.

HTTP/XML Response	Definition
sessionTimeoutInMins	The session timeout in minutes is the amount of minutes that must pass with no activity by the client before the CSM server discards the session as no longer active. Any access by the CSM client after a session has been timed out will be rejected and require the client to re-authenticate to gain access to the CSM server again. This attribute is only included if the user is successfully authenticated. Default is 15 minutes.

```
<xs:element name="loginResponse" type="LoginResponse"/>
<xs:complexType name="LoginResponse">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence>
                        <xs:element name="serviceVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="sessionTimeoutInMins" type="xs:positiveInteger" minOccurs="1"
maxOccurs="1"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Figure 14: LoginResponse XML Schema Method heartbeatCallback

An example of the Method heartbeatCallback is shown below. The fields in these messages are described in the table below.

```
URL:

https://csm-clienthost/heartbeatCallback

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<heartbeatCallbackRequest>
   <protVersion>1.0</protVersion>
</heartbeatCallbackRequest>
```

Figure 15: Method heartbeatCallback Example

Table 13: Method heartbeatCallback Elements and Attributes Description

HTTP Header/XML Argument	Definition
heartbeatCallbackRequest	The heatbeatCallback method is called by the CSM Server on the CSM client to ensure the csm client is still active
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session

HTTP Header/XML Argument	Definition
Returns	200 OK
	401 Unauthorized

Method specific errors:

Code	Description
7	Authentication Failure: Invalid username and/or password specified
8	Authorization Failure: Login session error. The asCookie contained in the request is incorrect. This error can also occur if the session is stale (invalidated/timed out). The session can be kept alive by using ping or heartbeat.
9	Maximum active session limit reached. No more sessions are allowed. (See API Administration options Section 1.13)
10	An error was encountered while connecting to a server resource. (Also applicable for logout method)
11 & 12	User input validation errors. Returned under any of the following conditions: • An error occured while parsing user input. • Heartbeat tag missing • Invalid callback URL specified or Callback URL is missing • Callback URL specified when heartbeat requested is false
14	Session Creation failed
16	Only HTTPS protocol allowed for callback url

Table 14: Login Method Error Codes

2.2.3 Method logout

The logout method notifies the CSM server that a previously authenticated CSM client is no longer requiring session access to the CSM server. The CSM client should logout from the CSM server if it does not intend to access methods on the CSM server within the session access timeout window.

2.2.3.1 Request

An example of the method logout request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/logout

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<logoutRequest>
      <protVersion>1.0</protVersion>
      <reqId>123</reqId>
</logoutRequest>
```

Figure 16: logout Request Example

Table 15: logout Request Elements and Attributes Descriptions

HTTP/XML Argument	Definition
logoutRequest	XML argument that logs out the CSM client from the CSM server
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="logoutRequest" type="LogoutRequest"/>
<xs:complexType name="LogoutRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
      </xs:complexContent>
</xs:complexType>
```

Figure 17: LogoutRequest XML Schema

2.2.3.2 Response

An example of the logout response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version=”1.0” encoding=”UTF-8”?>
      <logoutResponse>
         <protVersion>1.0</protVersion>
         <reqId>123</reqId>
      </logoutResponse>
```

Figure 18: logout Response Example

Table 16: logout Response Elements and Attributes Description

XML Response	Definition
logoutResponse	Returns the session information or failure

```
<xs:element name="logoutResponse" type="LogoutResponse"/>
<xs:complexType name="logoutResponse">
      <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
      </xs:complexContent>
</xs:complexType>
```

Figure 19: LogoutResponse XML Schema

2.2.4 Method: ping

The ping method maintains an active authenticated session from timing out or being discarded by the server. The CSM client should call the ping method before every sessionTimeoutInMins to ensure the CSM server does not discard its authenticated session. *Note: The authenticated session inactivity timeout is reset implicitly if an authentication session is used to make a call to the server.*

2.2.4.1 Request

An example of the ping request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/ping

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version=”1.0” encoding=”UTF-8”?>
   <pingRequest>
          <protVersion>1.0</protVersion>
          <reqId>123</reqId>
   </pingRequest>
```

Figure 20: ping Request Example

Table 17: ping Request Elements and Attributes Descriptions

HTTP/XML Argument	Definition
pingRequest	XML argument that pings the CSM server
HTTP Method	PUT
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="pingRequest" type="PingRequest"/>
<xs:complexType name="PingRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
      </xs:complexContent>
</xs:complexType>
```

Figure 21: PingRequest XML Schema

2.2.4.2 Response

An example of the ping response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version=”1.0” encoding=”UTF-8”?>
   <pingResponse>
          <protVersion>1.0</protVersion>
          <reqId>123</reqId>
   </pingResponse>
```

Figure 22: ping Response Example

Table 18: ping Response Elements and Attributes Description

XML Response	Definition
pingResponse	Returns the ping information

```
<xs:element name="pingResponse" type="PingResponse"/>
<xs:complexType name="PingResponse">
      <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
      </xs:complexContent>
</xs:complexType>
```

Figure 23: PingResponse XML Schema

3 CSM Configuration Service API

The configuration service provides access to read and write the configuration of network and CSM policy objects. The state of a policy may either be committed or deployed. It is possible that a committed policy is one that has not yet been deployed to a device. The configuration service API policy methods will return **committed policies only**. The policy commit and deployment to the device are two separate operations. It is possible that the current running device configuration does not reflect all policy configuration changes until they are deployed successfully. See the class definitions of BasePolicy and BasePolicyObject for details on how the configuration state is captured.

The **configState** attribute for BasePolicy and BasePolicyObject elements (section 3.1.1 and 3.1.2) indicates this state. If the config state is “committed” the device for which this config has been fetched has pending committed changes that **are not yet deployed**. If the config state is “deployed” then it means that **all committed changes have been deployed i.e. the CSM policy and the device are in sync.**

Selective Policy Management is a CSM feature that allows a CSM administrator to selectively manage policies in CSM. Data corresponding to any policy *not selected for management* will not be maintained in the CSM policy database. And such policy data will not be returned by this API. Please consider using the Utility Service API under such cases.

3.1 Object Model

The following sections describe the object model used by the CSM Configuration Service.

3.1.1 Base Policy

The CSM object model consists of two primary classes - a “Policy” class that denotes a specific policy such as AAA policy, Interface Policy, Firewall Access Rule etc. Policy classes can additionally reference “Policy Objects” (also called as Building Blocks) that are used to denote reusable objects such as network addresses, services, port lists etc. As an example, a “Firewall Policy” can define source and destinations addresses as a “Network Policy Objects”. Once defined policy objects are reusable and could be used by multiple policies.

All Policy classes extend from a “BasePolicy” class and all “Policy Objects” extend from a base class called as “BasePolicyObject”. The following class diagram denotes this relationship.

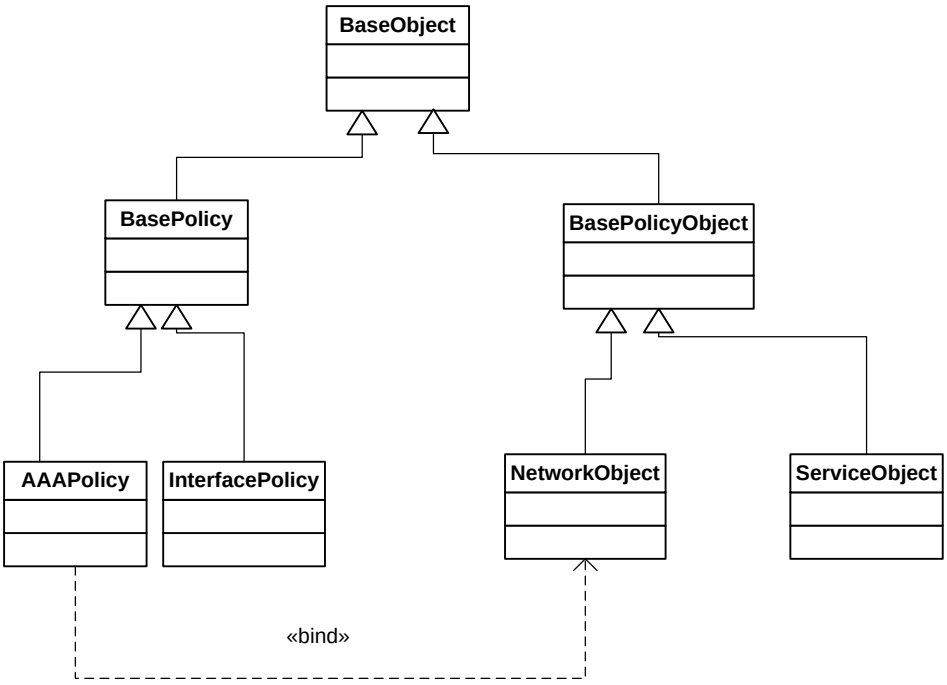


Figure 24: BasePolicy Class Inheritance

The BasePolicy class inherits from the BaseObject class including all attributes..

Attribute	Type	Comment
Type	String	It's a mandatory attribute of any policy that describes content of the policyExample – “AAAPolicy”.
isMandatoryAggregation	boolean	Relevant only for onion aggregation. Returns whether a policy is mandatory or not: mandatory policies will be pre-pended to parent and default ones will be appended.
orderId	Int	O-based ordering id. Returns index of the policy in the policy record.
Description	String	Policy Description (optional)
configState	Enumeration	The current state of the policy taken from { committed, deployed }

Table 19: BasePolicy Class Attributes

```
<xs:complexType name="BasePolicy">
    <xs:complexContent>
        <xs:extension base="BaseObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="orderId" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
                <xs:element name="isMandatoryAggregation" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="configState" type="ConfigurationState" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 25: BasePolicy XML Schema

The following sections denote the data content for important policy and policy object types.

3.1.2 BasePolicyObject

This is the base class for all Policy Objects which are reusable object definitions. Policy data like AAA policy, Firewall policy etc. maintain references to policy objects. There are multiple "types" of Policy Objects. Also a policy object can be "Global" or "Local". A global Policy object indicates that the object is a global definition that is referenced by policies on any device. A Policy Object "override" indicates a global policy object that is "overridden" for a specific device.

A set of policy objects may be "grouped" under a single policy object of the same "type". In some cases, a policy object could also reference a policy object of a totally different type (this is different from a "grouped" policy object which groups' policy objects of the same type).

The BasePolicyObject class inherits from BaseObject including all attributes.

Attribute	Type	Comment
Type	String	It's a mandatory attribute of any policy object that describes the type of the policy object. Example – "Network" or "Service".
Comment	String	Associated comment/description of this policy object (optional).
nodeGID	Object Identifier	The device ID if this is a "override" policy object. Set to -1 for Globals
isProperty	Boolean	A true value indicates that "overrides" for this Global Policy Object are allowed. A false indicates this global is not allowed to be "overridden"
subType	String	Sub-types applicable for "Network" and "Service" Policy Objects. For example "Host", "Address Range" are sub-types for a "Network" Policy object.
isGroup	Boolean	If true, this indicates whether this policy object is a "grouping" of other policy objects of the same type.
refGIDs	ObjectIdentifierList	Only applicable if "isGroup" is true. The list has the policy objects Id's this refers to.
configState	Enumeration	The current state of the policy object taken from { committed, deployed }

Table 20: BasePolicyObject Class Definition

Name override behavior - This is the name associated with the object. If the name is empty ("") then this refers to a internal policy object. All user defined policy objects must have a name. Internal policy objects are automatically created by the system in some cases. For example if a user provides a literal IP address in a rule (instead of a Policy Object), then a 'nameless' policy object is automatically created for the rule.

parentId = The parent Global Policy Object ID for which this Policy Object is an "override". For non-overrides, this is set to -1.

```
<xs:complexType name="BasePolicyObject" >
    <xs:complexContent>
        <xs:extension base="BaseObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="nodeGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="isProperty" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="subType" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="isGroup" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="refGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="configState" type="ConfigurationState" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 26: BasePolicyObject XML Schema

3.1.3 Policy Utility Classes

The following utility classes are defined for use across multiple policies:

- NetworkInterfaceObjectsRefs
    - Refers to a list of networks, interfaceRoles and ipv4 data strings
- NetworkObjectsRefs
    - Refers to a list of networks and ipv4 data strings
- IdentityUserGrpObjectsRefs
    - Refers to a list of identity user group objects GIDs, user names and user groups
- SecurityGrpObjectsRef
    - Refers to a a security name, security tag or a security object GID
- SecurityGrpObjectsRefs
    - Refers to a list of SecurityGrpObjectsRef
- NetworkObjectRefs
    - Refers to a network, ipv4 data string and interface keyword
- NetworkOrIPRef
    - Refers to either a host, network or a ipv4 data string

```
    <xs:complexType name="NetworkInterfaceObjectsRefs">
        <xs:sequence>
            <xs:element name="networkObjectGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="interfaceRoleObjectGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NetworkObjectsRefs">
        <xs:sequence>
            <xs:element name="networkObjectGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SecurityGrpObjectsRef">
      <xs:sequence>
        <xs:choice>
          <xs:element name="securityGrpObjectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
          <xs:element name="secName" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="secTag" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SecurityGrpObjectsRefs">
      <xs:sequence>
        <xs:element name="securityTag" type="SecurityGrpObjectsRef" minOccurs="1"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
<xs:complexType name="IdentityUserGrpObjectsRefs">
        <xs:sequence>
            <xs:element name="identityUserGrpObjectGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="userNameData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="userGroupData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NetworkObjectRefs">
        <xs:sequence>
            <xs:element name="networkObjectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="interfaceKeyword" type="xs:string" fixed="interface" minOccurs="0"
maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NetworkOrIPRef">
        <xs:choice>
            <xs:element name="hostOrNetworkObjectGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:choice>
    </xs:complexType>
```

Figure 27: Policy Utility Class XML Schema

3.1.4 PolicyObject Derived Classes

This section and sub-sections define the supported PolicyObject classes for this API.

3.1.4.1 NetworkPolicyObject

A NetworkPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A NetworkPolicyObject defines an IPv4 address, network or range.

Policy definitions reference the NetworkPolicyObject via the gid value. The inherited "**subtype"** attribute defines the type of IPv4 data contained. The allowable values for subtype for a NetworkPolicyObject are “Host”, “Network”, “Address Range”, “FQDN” and “Group”. The contents of a NetworkPolicyObject can also be “empty” in some cases when the inherited **isGroup** attribute is set to true (and subtype is “Group”). In such cases the NetworkPolicyObject is itself a container reference to “other Network Policy Objects”.

The list of gid values for such a PolicyObject is obtained from the **refs** inherited attribute. Also a “Group” NetworkPolicyObject can sometimes also contain multiple IPv4Data elements denoting literal IPv4 address, network or ranges. The combinations of data from the refs attribute references and the IPv4Data elements denote the complete group of addresses the policy object references.

Element	Type	Comment
ipv4Data	String	Defines a specific IPv4 data like address, range or network.
ipData	String	Defines a specific IP data like address, range or network.It can be both IPv4 and IPv6
fqdnData	Complex	Contains Fully Qualified Domain Name (FQDN) if this is a FQDN type NetworkPolicyObjects
fqdnData.value	String	The FQDN string
fqdnData.isIPv4Only	boolean	If true, the command generated and sent to the device contains the “v4” parameter.

Table 21: NetworkPolicyObject Class Definition

NOTE: From API Version 1.1 onward, a new tag called <ipData> has been added to the Network Object definition. Network objects referenced in legacy policies like DeviceAccessRuleFirewallPolicy will continue to use <ipv4Data> as these policies only reference IPv4 addresses. However, newer policies like DeviceAccessRuleUnifiedFirewallPolicy will use the <ipData> tag in the policy. This is because an <ipData> tag can contain both IPv4 and IPv6 addresses.

```
<xs:complexType name="NetworkPolicyObject">
    <xs:complexContent>
      <xs:extension base="BasePolicyObject">
        <xs:sequence>
          <xs:choice>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="ipData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          </xs:choice>
          <xs:element name="fqdnData" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="value" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="isIPv4Only" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

Figure 28: NetworkPolicyObject XML Schema

3.1.4.2 IdentityUserGroupPolicyObject

A IdentityUserGroupPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A IdentityUserGroupPolicyObject defines users and user groups. Policy definitions reference the IdentityUserGroupPolicyObject via the gid value.

Element	Type	Comment
userNameData	String	List of users in the group object.
userGroupData	String	List of user groups part of this group object.

Table 22: IdentityUserGroupPolicyObject Class Definition

```
<xs:complexType name=”IdentityUserGroupPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence>
                <xs:element name="userNameData" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="userGroupData" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 29: IdentityUserGroupPolicyObject XML Schema

3.1.4.3 PortListPolicyObject

A PortListPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A PortListPolicyObject defines an individual port or a group of ports. Policy definitions reference the PortListPolicyObject via the gid value. The contents of a PortListPolicyObject can also be "empty" in some cases when the inherited **isGroup** attribute is set to true. In such cases the PortListPolicyObject is itself a container reference to "other Port List Policy Objects". The list of gid values for such a PolicyObject is obtained from the **refs** inherited attribute.

In some cases the PortListPolicyObject can have a group of gid values from the **refs** attribute and *additionally* contain a definition for a single port list object. In such cases, the combination of references and the value inside the PortListPolicyObject element together constitute the groups of ports specified by the definition.

The following table defines the contents of the ServicePolicyObject:

Element	Type	Comment
startPort	PortIdentifier	Defines the starting port.
endPort	PortIdentifier	Defines the end port. Start port to end port specify the range of ports specified by this PolicyObject. If start and end Ports are equal then the definition corresponds to a "single" port.

Table 23: PortListPolicyObject Class Definition

```
<xs:complexType name="PortListPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:element name="startPort" type="PortIdentifier" minOccurs="1" maxOccurs="1"/>
                <xs:element name="endPort" type="PortIdentifier" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 30: PortListPolicyObject XML Schema

3.1.4.4 ServicePolicyObject

A ServicePolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A ServicePolicyObject defines services such as tcp, udp etc. Policy definitions reference the ServicePolicyObject via the gid value. The contents of a ServicePolicyObject can also be "empty" in some cases when the inherited **isGroup** attribute is set to true. In such cases the ServicePolicyObject is itself a container reference to "other Service Policy Objects". The list of gid values for such a PolicyObject is obtained from the **refs** inherited attribute.

In some cases the ServicePolicyObject can have a group of gid values from the **refs** attribute and *additionally* contain a definition for a single service. In such cases, the combination of references and the value inside the ServicePolicyObject element together constitute the groups of ports specified by the definition.

The following table defines the contents of the ServicePolicyObject:

Element	Type	Comment
protocol	String	A string defining a protocol such as tcp, udp etc.
sourcePort	Complex	Element container that holds the source port data. This is a choice type complex type so only one of the sub-elements will be defined.
sourcePort.port	Unsigned Int	The actual port value.
sourcePort.portRef	ObjectIdentifier	Gid reference to a PortListPolicyObject.
destinationPort	Complex	Element container that holds the destination port data. This is a choice type complex type so only one of the sub-elements will be defined.
destinationPort.port	Unsigned Int	The actual port value.
destinationPort.portRef	ObjectIdentifier	Gid reference to a PortListPolicyObject.
icmpMessage	String	ICMP Message content.

Table 24: ServicePolicyObject Class Definition

```
    <xs:complexType name="ServiceParameters">
        <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="protocol" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="sourcePort" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="port" type="xs:unsignedInt"/>
                        <xs:element name="portRefGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="destinationPort" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="port" type="xs:unsignedInt"/>
                        <xs:element name="portRefGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="icmpMessage" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ServicePolicyObject">
        <xs:complexContent>
            <xs:extension base="BasePolicyObject">
                <xs:sequence minOccurs="0" maxOccurs="1">
                    <xs:element name="serviceParameters" type="ServiceParameters" minOccurs="1"
maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 31: ServicePolicyObject XML Schema

3.1.4.5 InterfaceRolePolicyObject

An InterfaceRolePolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A InterfaceRolePolicyObject defines a pattern to denote an interface or group of interfaces like FastEthernet0, inside, outside etc.. The inherited gid attribute specifies a unique ID for an 'instance' of a InterfaceRole PolicyObject. Policy definitions reference the InterfaceRolePolicyObject via the gid value.

The following table defines the contents of the InterfaceRolePolicyObject:

Element	Type	Comment
pattern	String	Defines one or more interface role regex patterns. A"*" will match all interfaces.

Table 25: InterfaceRolePolicyObject Class Definition

```
<xs:complexType name="InterfaceRolePolicyObject">
        <xs:complexContent>
            <xs:extension base="BasePolicyObject">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="pattern" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 32: InterfaceRolePolicyObject XML Schema

3.1.4.6 TimeRangePolicyObject

A TimeRangePolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A TimeRangePolicyObject defines a time range and its associated recurrence. Policy definitions reference the TimeRangePolicyObject via the gid value. The contents of a TimeRangePolicyObject cannot be empty and there are also no "grouping" for TimeRangePolicyObject supported

The following table defines the contents of the TimeRangePolicyObject:

Element	Type	Comment
startTime	dateTime	Specifies the start time. If the start time is not specified, it means the time range specified by this range is "already started".
endTime	dateTime	Specifies the end time. If the end time is not specified, it means the time range specified by this range is "never ends".
recurrence	Complex type	A time range can have zero or more recurring ranges that qualify the recurrence for the time range. This is a choice type complex element and can contain either a "Day of the Week" based recurrence pattern or "Weekly" based recurrence pattern
recurrence.dayOfWeekInterval	Complex Type	Defines the "Day of the Week" based recurrence.
recurrence.dayOfWeekInterval.dayOfWeek	String	The dayOfWeek attribute defines days of the week tha pattern should be applicable. It can have any "one" of the following String values: • **Everyday**: For every day recurrence (OR) • **Weekday**: For Week day recurrence (OR) • **Weekends**: For Week end recurrence (OR) • A comma separated list of specific days of the week – Monday,Tuesday, .. etc.
recurrence.dayOfWeekInterval.startTime	Time	Specifies the start time
recurrence.dayOfWeekInterval.endTime	Time	Specifies the end time.
recurrence.weeklyInterval	Complex Type	Defines the "Weekly" based recurrence.
recurrence.weekInterval.startDay	String	Defines the start day i.e "Monday", "Tuesday" ….
recurrence.weeklyInterval.startTime	Time	Specifies the start time
recurrence.weeklyInterval.endDay	String	Defines the end day i.e "Monday", "Tuesday" ….
recurrence.weeklyInterval.endTime	Time	Specifies the end time

Table 26: TimeRangePolicyObject Class Definition

```
<xs:complexType name="TimeRangePolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="startTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
                <xs:element name="endTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
                <xs:element name="recurrence" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:choice>
                            <xs:element name="dayOfWeekInterval">
                                <xs:complexType>
                                    <xs:sequence minOccurs="1" maxOccurs="1">
                                        <xs:element name="dayOfWeek" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                        <xs:element name="startTime" type="xs:time"
minOccurs="1" maxOccurs="1"/>
                                        <xs:element name="endTime" type="xs:time"
minOccurs="1" maxOccurs="1"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="weeklyInterval">
                                <xs:complexType>
                                    <xs:sequence minOccurs="1" maxOccurs="1">
                                        <xs:element name="startDay" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                        <xs:element name="startTime" type="xs:time"
minOccurs="1" maxOccurs="1"/>
                                        <xs:element name="endDay" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                        <xs:element name="endTime" type="xs:time"
minOccurs="1" maxOccurs="1"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 33: TimeRangePolicyObject XML Schema

3.1.4.7 SLA Monitor Policy Object

A SLAMonitorPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A SLAMonitorPolicyObject defines a SLA Monitoring policy. Policy definitions reference the SLAMonitorPolicyObject via the gid value.

The following table defines the contents of the SLAMonitorPolicyObject:

Element.Sub Element	Type	Comment
slaId	int	The ID number of the SLA operation.
interface	ObjectIdentifier	The source interface for all ICMP echo requests sent to the monitored address to test its availability. Enter the name of an interface or interface role, or click **Select** to select an it from a list or to create a new interface role.
monitoredAddress	string	The IP address that is being monitored for availability by the SLA operation.
dataSizeInBytes	int	The size of the ICMP request packet payload, in bytes.
thresholdInMilliSeconds	int	The amount of time that must pass after an ICMP echo request before a rising threshold is declared, in milliseconds.
timeoutInMilliSeconds	int	The amount of time that the SLA operation waits for a response to the ICMP echo requests, in milliseconds.
frequencyInSeconds	int	The frequency of ICMP echo request transmissions, in seconds.
toS	int	The type of service (ToS) defined in the IP header of the ICMP request packet. Values range from 0 to 255. The default is 0.
numberOfPackets	int	The number of packets that are sent. Values range from 1 to 100. The default is 1 packet.

Table 27: SLAMonitorPolicyObject Class Definition

3.1.4.8 Standard ACE Policy Object

A StandardACEPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A StandardACEPolicyObject defines a standard IP access control entry. Policy definitions reference the StandardACEPolicyObject via the gid value.

Element.Sub Element	Type	Comment
networkGID	ObjectIdentifier	The source or destination of the traffic.
doLogging	boolean	Whether to create log entries when traffic meets the entry criteria.

permit	boolean	Indicates the action to be taken when a match is found

Table 28: StandardACEPolicyObject Class Definition

```
<xs:complexType name="StandardACEPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="networkGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="doLogging" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                <xs:element name="permit" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 34: StandardACEPolicyObject XML Schema

3.1.4.9 Extended ACE Policy Object

An ExtendedACEPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. An ExtendedACEPolicyObject defines an extended access control entry. Policy definitions reference the ExtendedACEPolicyObject via the gid value.

Element.Sub Element	Type	Comment
sourceGID	ObjectIdentifier	The source of the traffic.
destinationGID	ObjectIdentifier	Traffic destination.
serviceGID	ObjectIdentifier	The service that defines the type of traffice to act upon
doLogging	String	Contains value “true” if logging is enabled for PIX, ASA, FWSM devices or “false” otherwise. If logInterval and logLevel elements are not specified then it means “Default Logging” is enabled.
logInterval	String	Specifies the Logging Interval in seconds, if this is specified it means, “per ACE Logging is Enabled”
logLevel	String	Specifies the Logging Level – one of “Emergency”, “Alert”, “Critical”, “Error”, “Warning”, “Notification”, “Informational” or “Debugging. Iif this is specified it means, “per ACE Logging is Enabled”
logOption	String	Used to specify IOS logging. Contains “log” if IOS logging is enabled. Contains “log-input” if IOS Logging is enabled and Log Input is also enabled for IOS devices.
permit	boolean	True if this is a permit ACE, false for deny.

Table 29: ExtendedACEPolicyObject Class Definition

```
<xs:complexType name="ExtendedACEPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence>
                <xs:element name="sourceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                <xs:element name="destinationGID" type=" ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="serviceGID" type=" ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="doLogging" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="logInterval" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="logLevel" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="logOption" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="permit" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 35: ExtendedACEPolicyObject XML Schema

3.1.4.10 ACL Policy Object

Element.Sub Element	Type	Comment
references	Complex Type	List of references to ACE's
references .sequenceNumber	Unsigned int	Sequence number of this entry
references .aclObjectReferenceGID	ObjectIdentifier	Reference to the ACE policy object
References.aceReferenceGID	ObjectIdentifier	Reference to the standard/extended ACE policy object

Table 30: ACLPolicyObject Class Definition

```
<xs:complexType name="ACLPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="references" minOccurs="1" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence minOccurs="1" maxOccurs="1">
                            <xs:element name="sequenceNumber" type="xs:unsignedInt"
minOccurs="1" maxOccurs="1"/>
                            <xs:choice>
                                <xs:element name="aclObjectReferenceGID"
type="ObjectIdentifier"/>
                                <xs:element name="aceReferenceGID" type="ObjectIdentifier"/>
                            </xs:choice>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 36: ACLPolicyObject XML Schema

3.1.4.11 SecurityGroupPolicyObject

A SecurityGroupPolicyObject extends from the **BasePolicyObject** class and inherits all its attributes. A SecurityGroupPolicyObject defines a security tag or name.

Policy definitions reference the SecurityGroupPolicyObject via the gid value. The contents of a SecurityGroupPolicyObject can also be "empty" in some cases when the inherited **isGroup** attribute is set to true In such cases the SecurityGroupPolicyObject is itself a container reference to "other Security Group Policy Objects".

The list of gid values for such a PolicyObject can be obtained from the **refs** inherited attribute. Also a "Group" SecurityGroupPolicyObject can sometimes also contain multiple secuTag elements denoting literal security tag/name or contained policy object. The complete content of the security policy object can be obtained from the securityTag elements..

This object is available beginning with Version 1.1 of the API.

Element	Type	Comment
securityTag	Complex(SecurityGrpObjectsRef)	Defines a security tag/name or GID of the referred policy object..
securityTag .securityGrpObjectGID	ObjectIdentifier	Defines the GID of the refererred security group policy object.
securityTag .secName	String	Defines a security tag/
securityTag .secTag	String	Defines a security tag/name

Table 31: SecurityGroupPolicyObject() Class Definition

```
<xs:complexType name="SecurityGroupPolicyObject">
  <xs:complexContent>
    <xs:extension base="BasePolicyObject">
      <xs:sequence>
        <xs:element name="securityTag" type="SecurityGrpObjectsRef" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 37: SecurityGroupObject XML Schema

```
<xs:complexType name="SecurityGrpObjectsRef">
  <xs:sequence>
    <xs:choice>
      <xs:element name="securityGrpObjectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
      <xs:element name="secName" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="secTag" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Figure 38: SecurityGrpObjectsRef XML Schema

3.1.5 Policy Derived Classes

This section and the following sub-sections define supported Policy classes over this API

3.1.5.1 DeviceAccessRuleFirewallPolicy

A DeviceAccessRuleFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a DeviceAccessRuleFirewallPolicy denotes a single access control entry. The orderId attribute from the BasePolicy class defines the ordering of these rules.

A DeviceAccessRuleFirewallPolicy may reference NetworkPolicyObject, ServicePolicyObject, IdentityUserGroupPolicyObject or an InterfaceRolePolicy Object objects.

The **sources** and **destination** elements may contain **a combination** of any of the following:

- **networkObjectGIDs**: This includes one or more GID references to a Network Policy Object
- **interfaceRoleObjectGIDs**: This includes one or more GID references to a Interface Role Policy Object
- **ipv4Data**: One or more literal IPv4 addresses

It is possible to specify a destination element referring two Network Policy Objects and also including a literal address such as 1.1.1.1/32.

The **interfaceRoleObjectIDs** specified *outside* the sources and destination sub-elements specifies the interface on which the ACL is applied.

The following table defines the contents of a DeviceAccessRuleFirewallPolicy:

Element	Type	Comment
isEnabled	Boolean	True if the rule is enabled, false otherwise.
direction	String	in or out.
permit	boolean	True indicates a permit and false indicates a deny
interfaceRoleObjectIDs	ObjectIdentifierList	A list of ObjectIdentifier IDs that reference a set of InterfaceRole Policy Objects. The ID links to the gid attribute of the corresponding InterfaceRole object
users	ComplexType	Contain user and user groups for whom the rule is applicable (applicable only for ASA device versions greater than or equal to 8.4(2))
users. identityUserGrpObject GIDs	ObjectIdentifierList	Reference to list of IdentityUserGroupPolicyObject object GIDs.
users. userNameData	String	List of users.
users. userGroupData	String	List of user groups.
sources	ComplexType	Container for source network and interface roles.
sources.networkObject GIDs	ObjectIdentifierList	A list of ObjectIdentifier IDs that reference a set of Network Policy Objects. The ID links to the gid attribute of the corresponding Network object
sources.interfaceRoleO bjectGIDs	ObjectIdentifierList	A list of ObjectIdentifier IDs that reference a set of InterfaceRole Policy Objects. The ID links to the gid attribute of the corresponding InterfaceRole object

Element	Type	Comment
sources.ipv4Data	String	Multiple IPv4Data elements containing either IPv4 host, network or ranges.
destination	ComplexType	Container for destination network and interface roles.
destination.networkObjectGIDs	ObjectIdentifierList	A list of ObjectIdentifier IDs that reference a set of Network Policy Objects. The ID links to the gid attribute of the corresponding Network object
destination.interfaceObjectGIDs	ObjectIdentifierList	A list of ObjectIdentifier IDs that reference a set of InterfaceRole Policy Objects. The ID links to the gid attribute of -the corresponding InterfaceRole object
destination.ipv4Data	String	Multiple IPv4Data elements containing either IPv4 host, network or ranges.
serviceObjectIDs	ObjectIdentifierList	A list of ObjectIdentifier IDs that reference a set of Service Policy Objects. The ID links to the gid attribute of the corresponding Service object
serviceParameters	ServiceParameters	Multiple service data elements that can store service data including protocol and port details
serviceParameters.protocol	String	A string defining a protocol such as tcp, udp etc.
serviceParameters.sourcePort	ComplexType	Element container that holds the source port data. This is a choice type complex type so only one of the sub-elements will be defined.
serviceParameters.sourcePort.port	String	The actual port value. Can also be a range specified in the form <start port>-<end port> example: 1-65535
serviceParameters.sourcePort.portRef	ObjectIdentifier	Gid reference to a PortListPolicyObject.
serviceParameters.destinationPort	ComplexType	Element container that holds the destination port data. This is a choice type complex type so only one of the sub-elements will be defined.
serviceParameters.destinationPort.port	String	The actual port value. Can also be a range specified in the form <start port>-<end port> example: 1-65535
serviceParameters.destinationPort.portRef	ObjectIdentifier	Gid reference to a PortListPolicyObject.
serviceParameters.icmpMessage	String	ICMP Message content.
logOptions	ComplexType	Root element that stores the logging options associated with a rule
logOptions.isFirewallLoggingEnabled	Boolean	Indicates whether logging is enabled for PIX, ASA, FWSM rules
logOptions.isDefaultLogging	Boolean	Whether default logging is enabled. This is applicable for PIX, ASA and FWSM devices and only if isFirewallLoggingEnabled is true
logOptions.loggingInter	Int	Specifies the logging interval in seconds. This is applicable for

Element	Type	Comment
val		PIX, FWSM and ASA and if isDefaultLogging is false or not specified. (Applicable for per ACE logging)
logOptions.loggingLevel	String	Specifies a logging level – “Emergency”, “Alert”, “Critical”, “Error” “Warning”, “Notification”, “Informational” or “Debugging”. This is applicable for PIX, FWSM and ASA and if isDefaultLogging is false or not specified. (Applicable for per ACE logging)
logOptions. isIOSLoggingEnabled	Boolean	Set to true is logging for IOS devices is enabled. Only applicable for IOS
logOptions.isLogInput	Boolean	True if log input is enabled. This is applicable for IOS devices and only if isIOSLoggingEnabled is true.
iosOptions	String	IOS Options allowed values are “none”, “Fragment” and “Established”.
timeRangeObjectId	ObjectIdentifier	References a TimeRangePolicyObject if a time range is specified for this rule.

Table 32: DeviceAccessRuleFirewallPolicy Class Definition

See the XML schema at the end of this document for the XML schema for this class.

3.1.5.1.1 Policy Config Device Response Example

```
<?xml version="1.0" encoding="utf-8"?>
<n:policyConfigDeviceResponse xmlns:n="csm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" >
    <policy>
        <deviceAccessRuleFirewallPolicy>
            <gid>00000000-0000-0000-0000-000000000889</gid>
            <name>string</name>
            <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
            <updatedByUser>admin</updatedByUser>
            <lastCommitTime>2012-12-05T09:18:39.371Z </lastCommitTime>
            <activityName>admin_05.Dec.2012_01.17.10</activityName>
            <type>Access Rule</type>
            <orderId>0</orderId>
            <isMandatoryAggregation>false</isMandatoryAggregation>
            <description/>
            <isEnabled>true</isEnabled>
            <direction>in</direction>
            <permit>true</permit>
            <interfaceRoleObjectGIDs>00000000-0000-0000-0000-
000000000123</interfaceRoleObjectGIDs>
            <sources>
                <networkObjectGIDs>00000000-0000-0000-0000-
000000000124</networkObjectGIDs>
                <interfaceRoleObjectGIDs>00000000-0000-0000-0000-
000000000123</interfaceRoleObjectGIDs>
                <ipv4Data>1.1.1.1/32</ipv4Data>
            </sources>
            <destinations>
                <networkObjectGIDs>00000000-0000-0000-0000-
000000000125</networkObjectGIDs>
                <interfaceRoleObjectGIDs>00000000-0000-0000-0000-
000000000123</interfaceRoleObjectGIDs>
                <ipv4Data>2.2.2.2</ipv4Data>
            </destinations>
            <services>
                <serviceObjectGIDs>00000000-0000-0000-0000-
000000000126</serviceObjectGIDs>
                <serviceParameters>
                    <protocol>tcp</protocol>
                    <sourcePort>
                        <port>80</port>
                    </sourcePort>
                    <destinationPort>
                        <port>80</port>
                    </destinationPort>
                </serviceParameters>
            </services>
            <logOptions>
                <isFirewallLoggingEnabled>false</isFirewallLoggingEnabled>
            </logOptions>
            <iosOptions>None</iosOptions>
```

```
            <timeRangeObjectGID>00000000-0000-0000-0000-
000000000127</timeRangeObjectGID>
        </deviceAccessRuleFirewallPolicy>
    </policy>
    <policyObject>
        <networkPolicyObject>
            <gid>00000000-0000-0000-0000-000000000124</gid>
            <name>mySource</name>
            <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
            <type>Network</type>
            <comment/>
            <isProperty>false</isProperty>
            <subType>Host</subType>
            <isGroup>false</isGroup>
            <refGIDs/>
            <ipv4Data>1.1.2.2</ipv4Data>
        </networkPolicyObject>
        <networkPolicyObject>
            <gid>00000000-0000-0000-0000-000000000125</gid>
            <name>myDest</name>
            <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
            <type>Network</type>
            <comment/>
            <isProperty>false</isProperty>
            <subType>Host</subType>
            <isGroup>false</isGroup>
            <refGIDs/>
            <ipv4Data>1.1.3.3</ipv4Data>
        </networkPolicyObject>
        <portListPolicyObject>
            <gid>00000000-0000-0000-0000-000000002782</gid>
            <name>MyPortList</name>
            <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
            <type>PortList</type>
            <comment/>
            <isProperty>false</isProperty>
            <subType>Host</subType>
            <isGroup>false</isGroup>
            <refGIDs/>
            <startPort>514</startPort>
            <endPort>514</endPort>
        </portListPolicyObject>
        <servicePolicyObject>
            <gid>00000000-0000-0000-0000-000000000126</gid>
            <name>string</name>
            <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
            <type>Service</type>
            <comment/>
            <isProperty>false</isProperty>
            <subType/>
            <isGroup>false</isGroup>
            <refGIDs/>
            <serviceParameters>
                <protocol>udp</protocol>
```

```
                    <sourcePort>
                        <portRefGID>00000000-0000-0000-0000-000000002782</portRefGID>
                    </sourcePort>
                    <destinationPort>
                        <port>514</port>
                    </destinationPort>
                    <icmpMessage>string</icmpMessage>
                </serviceParameters>
                    </servicePolicyObject>
                <interfaceRolePolicyObject>
                    <gid>00000000-0000-0000-0000-000000000123</gid>
                    <name>FA1</name>
                    <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
                    <type>InterfaceRole</type>
                    <comment/>
                    <isProperty>false</isProperty>
                    <subType/>
                    <isGroup>false</isGroup>
                    <refGIDs/>
                    <pattern>FastEthernet0/0</pattern>
                </interfaceRolePolicyObject>
                <timeRangePolicyObject>
                    <gid>00000000-0000-0000-0000-000000000127</gid>
                    <name>string</name>
                    <lastUpdateTime>1697-02-01T00:00:00Z</lastUpdateTime>
                    <type>TimeRange</type>
                    <comment/>
                    <isProperty>false</isProperty>
                    <subType/>
                    <isGroup>false</isGroup>
                    <refGIDs/>
                    <startTime>1697-02-01T00:00:00Z</startTime>
                    <endTime>1697-02-01T00:00:00Z</endTime>
                    <recurrence>
                        <weeklyInterval>
                            <startDay>Monday</startDay>
                            <startTime>13:20:00-05:00</startTime>
                            <endDay>Friday</endDay>
                            <endTime>13:20:00-05:00</endTime>
                        </weeklyInterval>
                    </recurrence>
                </timeRangePolicyObject>
        </policyObject>
</n:policyConfigDeviceResponse>
```

3.1.5.2 DeviceAccessRuleUnifiedFirewallPolicy

A DeviceAccessRuleUnifiedFirewallPolicy extends from the base DeviceAccessRuleFirewallPolicy class and inherits all its attributes. An instance of a DeviceAccessRuleUnifiedFirewallPolicy denotes a single unified access control entry. The orderId attribute from the BasePolicy class defines the ordering of these rules.

A DeviceAccessRuleUnifiedFirewallPolicy mayadditionally reference SecurityGroupPolicyObject,

The **sourceSG** and **destinationSG** elements SecurityGrpObjectsRefs which is a list of SecurityGrpObjectsRef. Refer to Sec 3.1.4.11 on SecurityGrpObjectsRef

The XML content contains all the attributes of the base except for IOS options like logging, fragment and established.

This policy is available beginning with Version 1.1 of the API.

Element	Type	Comment
sourceSG	Complex(SecurityGrpObjectsRefs)	Defines security tags or objects for the source of the incoming packet.
destinationSG	Complex(SecurityGrpObjectsRefs)	Defines security tags or objects for the destination of the incoming packet

Table 33: DeviceAccessRuleUnifiedFirewallPolicy Class Definition

```
<xs:complexType name="DeviceAccessRuleUnifiedFirewallPolicy">
   <xs:complexContent>
      <xs:extension base="DeviceAccessRuleFirewallPolicy">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="sourceSG" type="SecurityGrpObjectsRefs" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinationSG" type="SecurityGrpObjectsRefs" minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
</xs:complexContent>
</xs:complexType>
```

Figure 37: DeviceAccessRuleUnifiedFirewallPolicy XML Schema

3.1.5.3 FirewallACLSettingsPolicy

Element.Sub Element	Type	Comment
aclName	String	The name of the ACL
interfaceGID	ObjectIdentifier	Interface to which this ACL applies.
trafficDirection	String	The direction of the traffic to which the ACL applies. The values are In or Out
useUserDefinedACLName	boolean	Boolean option to use CSM defined ACL names or user defined ACL name.
enablePerUserDownloadable ACLs	boolean	Boolean option to enable per user downloadable ACLs
enableObjectGroupSearch	boolean	Boolean option to enable object group search
enableAccessListCompilation	boolean	Boolean option to enable Access List compilation

Table 34: FirewallACLSettingsPolicy

```
<xs:complexType name="FirewallACLSettingsPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <!-- aclName is mandatory if request has Global interface or if user
                    defined acl is set as true. if useUserDefinedACLName defined aclname is true
                    then aclName will be ignored -->
                <xs:element name="aclName" type="xs:string" minOccurs="0"
                    maxOccurs="1" />
                <xs:element name="interfaceGID" type="ObjectIdentifier"
                    minOccurs="0" maxOccurs="1" />
                <xs:element name="trafficDirection" minOccurs="1"
                    maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="In" />
                            <xs:enumeration value="Out" />
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="useUserDefinedACLName" type="xs:boolean"
                    minOccurs="1" maxOccurs="1" />
                <xs:element name="enablePerUserDownloadableACLs" type="xs:boolean"
                    minOccurs="1" maxOccurs="1" />
                <xs:element name="enableObjectGroupSearch" type="xs:boolean"
                    minOccurs="1" maxOccurs="1" />
                <xs:element name="enableAccessListCompilation" type="xs:boolean"
                    minOccurs="0" maxOccurs="1" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 38: FirewallACLSettingsPolicy

3.1.5.4 DeviceStaticRoutingFirewallPolicy

Element.Sub Element	Type	Comment
FwStaticRoutePolicy	Complex Type	
interfaceGID	ObjectIdentifier	Interface to which this static route applies.
networks	Complex Type	Destination network(s). You can provide one or more IP address/netmask entries, one or more Networks/Hosts objects, or a combination of both; separate the entries with commas. Use 0.0.0.0 to specify a default route. The 0.0.0.0 IP address can be abbreviated as 0.
networks.networkObjectIdentifierGIDs	ObjectIdentifierList	Value as reference to Policy Object
networks.ipv4Data	string	Value as raw string
Gateway	Complex Type	The gateway router, which is the next hop for this route.
gateway.hostObjectIdentifierGID	ObjectIdentifier	Reference to Policy Object
gateway.ipv4Data	string	Raw string value for gateway IP address
Metric	unsignedInt	The number of hops to the destination network. Valid values range from 1 to 255; the default value is 1.
Tunneled	boolean	Indicates whether this is a tunnel route.
slaMonitorGID	ObjectIdentifier	Name of an SLA (service level agreement) object that defines the monitoring policy.

Table 35: DeviceStaticRoutingFirewallPolicy

```
<xs:complexType name="DeviceStaticRoutingFirewallPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="networks" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence minOccurs="1" maxOccurs="1">
                                <xs:element name="networkObjectIdentifierGIDs"
type="ObjectIdentifierList" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="ipv4Data" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="gateway" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence minOccurs="1" maxOccurs="1">
                                <xs:element name="hostObjectIdentifierGID" type="ObjectIdentifier"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="ipv4Data" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="metric" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="tunnelled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="slaMonitorGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1"/>

                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 39: DeviceStaticRoutingFirewallPolicy

3.1.5.5 DeviceStaticRoutingRouterPolicy

Element.Sub Element	Type	Comment
destinationNetwork	Complex Type	Container of elements for destination network configuration.
destinationNetwork.useAsDefaultRoute	boolean	Indicates whether the static route is the default route for unknown packets being forwarded by this router.
destinationNetwork.prefix	Complex Type	The destination IP address of the static route.
destinationNetwork.prefix.networkObjectReferenceGID	ObjectIdentifier	The destination IP address of the static route as Object Reference.
destinationNetwork.prefix.ipV4Data	string	The destination IP address of the static route in string format.
Forwarding	Complex Type	
fowarding.forwardingInterfaceGID	ObjectIdentifier	The interface name associated with the gateway router that is the next hop address for this router.
fowarding.forwardingIPAddress	Complex Type	The IP address associated with the gateway router that is the next hop address for this router.
fowarding.forwardingIPAddress.hostObjectReferenceGID	ObjectIdentifier	IP Address as object reference
fowarding.forwardingIPAddress.ipv4HostAddress	string	IP Address as string value
distanceMetric	unsignedInt	The number of hops from the gateway IP to the destination. The metric determines the priority of this route. The fewer the hops, the higher the priority assigned to the route, based on lower costs. When two routing entries specify the same network, the entry with the lower metric (that is, the higher priority) is selected.
isPermanentRoute	boolean	Indicates whether the static route is defined as a permanent route, which means that it will not be removed even if the interface is shut down or if the router is unable to communicate with the next router.

Table 36: DeviceStaticRoutingRouterPolicy

```
<xs:complexType name="DeviceStaticRoutingRouterPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="destinationNetwork" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:choice>
                                <xs:element name="useAsDefaultRoute" type="xs:boolean"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="prefix" minOccurs="1" maxOccurs="1">
                                    <xs:complexType>
                                        <xs:choice>
                                            <xs:element name="networkObjectReferenceGID"
type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                                            <xs:element name="ipV4Data" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                        </xs:choice>
                                    </xs:complexType>
                                </xs:element>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="fowarding" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:choice>
                                <xs:element name="forwardingInterfaceGID" type="ObjectIdentifier"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="forwardingIPAddress" minOccurs="1"
maxOccurs="1">
                                    <xs:complexType>
                                        <xs:choice>

    <xs:element name="hostObjectReferenceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                                            <xs:element name="ipv4HostAddress"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                                        </xs:choice>
                                    </xs:complexType>
                                </xs:element>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="distanceMetric" type="xs:unsignedInt" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="isPermanentRoute" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 39: DeviceStaticRoutingRouterPolicy

3.1.5.6 DeviceBGPRouterPolicy

Element.Sub Element	Type	Comment
asNumber	unsignedLong	The number of the autonomous system in which the router is located. Valid values range from 1 to 65535. This number enables a BGP routing process.
Networks	Complex Type	The networks associated with the BGP route.
networks.networkObjectIdentifierGIDs	ObjectIdentifierList	Value as reference to Policy Object
networks.ipv4Data	string	Value as string
Neighbors	Complex Type	The *internal* neighbors (those located in the same AS as the router) and *external* neighbors (located in different ASs) of the router.
neighbors.ipAddress	Complex Type	Neighbor address
neighbors.ipAddress.networkObjectGIDs	ObjectIdentifierList	Address as reference to policy object list
neighbors.ipAddress.ipv4Data	string	Address as string value
neighbors.asNumber	unsignedLong	AS number of the neighbor
autoSummary	boolean	Enables auto summary. When a subnet is redistributed from an IGP (such as RIP, OSPF or EIGRP) into BGP, this BGP Version 3 features injects only the network route into the BGP table. Automatic summarization reduces the size and complexity of the routing table that the router must maintain.
synchronization	boolean	Enables Synchronization. When selected, synchronization is enabled. Use this feature to ensure that all routers in your network are consistent about the routes they advertise. Synchronization forces BGP to wait until the IGP propagates routing information across the AS. When deselected, synchronization is disabled. You can disable synchronization if this router does not pass traffic from a different AS to a third AS, or if all the routers in the AS are running BGP. Disabling this feature has the benefit of reducing the number of routes the IGP must carry, which improves convergence times. This is the default.
logNeighbor	boolean	Enables the logging of messages that are generated when a BGP neighbors resets, connects to the network, or is disconnected.
redistributionEntry	Complex Type	Redistribution settings when performing redistribution into a BGP autonomous system (AS). Can be multiple.
redistributionEntry.protocol	Complex Type	The protocol that is being redistributed.
redistributionEntry.protocol.st	String	Choice for protocol. Allowable values “IP” or “OSI”

atic		
redistributionEntry.protocol.connected	String	Choice for protocol
redistributionEntry.protocol.rip	String	Choice for protocol
redistributionEntry.protocol.eigrp	Complex Type	Choice for protocol
redistributionEntry.protocol.eigrp.asNumber	unsignedInt	The AS number if EIGRP is selected.
redistributionEntry.protocol.ospf	Complex Type	Choice of protocol
redistributionEntry.protocol.ospf.processId	UnsignedInt	The process number if EIGRP is selected.
redistributionEntry.protocol.ospf.match	String	Multiple Match criteria. Takes one of the allowable values for each match criterta : • **Internal** : Routes internal to the autonomous system (AS) are redistributed. • **External1**: Type 1 routes external to the AS are redistributed. • **External2** : Type 2 routes external to the AS are redistributed. • **NSSAExternal1** : Type 1 routes external to a not-so-stubby area (NSSA) are redistributed. • **NSSAExternal2** : Type 2 routes external to an NSSA are redistributed.
redistributionEntry.metric	unsignedLong	The value that determines the priority of the redistributed route.

Table 37: DeviceBGPRouterPolicy Class Definition

See the XML schema at the end of this document for the XML schema for this class.

3.1.5.7 InterfaceNATRouterPolicy

An InterfaceNATRouterPolicy extends from the BasePolicy class and inherits all its attributes. An instance of an InterfaceNATRouterPolicy specifies one NAT inside and outside interface.

The InterfaceNATRouterPolicy can reference an InterfaceRole PolicyObject.

The following table defines the contents of an InterfaceNATRouterPolicy:

Element. Sub Element	Type	Comment
interfaceGID	ObjectIdentifier	An ObjectIdentifier ID that references an InterfaceRole Policy Objects denoting the inside or outside interfaces. The ID links to the gid attribute of the corresponding InterfaceRole object.
isNatInside	boolean	A boolean which indicates whether this is a NAT inside interface (true) or outside interface (false)

Table 38: InterfaceNATRouterPolicy Class Definition

```
<xs:complexType name="InterfaceNATRouterPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="isNatInside" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 40: InterfaceNATRouterPolicy XML Schema

3.1.5.8 InterfaceNATStaticRulesRouterPolicy

An InterfaceNATStaticRulesRouterPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of an InterfaceNATStaticRulesRouterPolicy denotes static NAT rules.

The InterfaceNATStaticRulesRouterPolicy can reference a Network PolicyObject and an InterfaceRole Policy Object. The base orderId attribute defines the ordering of the static rules.

The following table defines the contents of a InterfaceNATStaticRulesRouterPolicy:

Element. Sub Element	Type	Comment
staticRuleType	Enumeration	The type of local address to be translated by this static rule: • "Static Host" – A single host requiring static address translation. • "Static Network" – A subnet requiring static address translation. • "Static Port" – A single port requiring static address translation. If you select this option, you must define the

Element. Sub Element	Type	Comment
		Port Redirection parameters.
Original	Complex Type	A complex type element that identifies an IP address, or a network/host object representing the address(es) to be translated.
original.ipv4Data	String	A literal IP address.
original.networkObjectGID	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
translated	Complex Type	A complex type element containing the addresses to which the Original Addresses are translated. This can contain either a specific IP address/network object or can be used to specify an interface. If the interface is specified, the IP addresses assigned to the interface is used as a translated address.
translated.originalIP	Complex Type	Complex Type that specifies an IP data or network policy object.
translated.originalIP.ipv4Data	String	A literal IP Address.
translated.originalIP.networkObjectGID	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
translated.interfaceGID	Object identifier	An ObjectIdentifier ID that references a InterfaceRole Policy Object.
portRedirection	Complex Type	A complex type that specifies port information for the address translations. These parameters are available only when Static Port is the chosen rule type.
portRedirection.protocol	String	The communications protocol used for these ports: TCP or UDP.
portRedirection.localPort	unsignedInt	The port number on the source network. Valid values range from 1 to 65535.
portRedirection.globalPort.	unsignedInt	The port number on the destination network that the router is to use for this translation. Valid values range from 1 to 65535.
Settings	Complex Type	An optional complex type element that contains advanced options.
settings.noAlias	boolean	Disable automatic aliasing for the global IP address translation if true
settings.noPayload	boolean	Prohibit an embedded address or port in the payload from being translated if true.
settings.createExtTransEntry	boolean	Extended translation entries (addresses and ports) are created in the translation table if true.

Table 39: InterfaceNATStaticRulesRouterPolicy Class Definition

```
<xs:complexType name="InterfaceNATStaticRulesRouterPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="staticRuleType" minOccurs="1" maxOccurs="1">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="Static Host"/>
                                <xs:enumeration value="Static Network"/>
                                <xs:enumeration value="Static Port"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="original" type="NetworkOrIPRef" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="translated" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:choice>
                                <xs:element name="originalIP" type="NetworkOrIPRef"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="interfaceGID" type="ObjectIdentifier"
minOccurs="1" maxOccurs="1"/>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="portRedirection" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="protocol" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                                <xs:element name="localPort" type="xs:unsignedInt" minOccurs="1"
maxOccurs="1"/>
                                <xs:element name="globalPort" type="xs:unsignedInt" minOccurs="1"
maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="settings" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="noAlias" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                                <xs:element name="noPayload" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                                <xs:element name="createExtTransEntry" type="xs:boolean"
minOccurs="1" maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 41: InterfaceNATStaticRulesRouterPolicy XML Definition

3.1.5.9 InterfaceNATDynamicRulesRouterPolicy

An InterfaceNATDynamicRulesRouterPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a InterfaceNATDynamicRulesRouterPolicy specifies NAT dynamic rules.

The InterfaceNATDynamicRulesRouterPolicy can reference an ACL PolicyObject and an InterfaceRole Policy Object. The base orderId attribute defines the ordering of the dynaimc rules

The following table defines the contents of an InterfaceNATDynamicRulesRouterPolicy:

Element. Sub Element	Type	Comment
trafficFlowAclObjectGID	ObjectIdentifier	References an access control list (ACL) Policy object GID whose entries define the addresses requiring dynamic translation.
translatedAddress	Complex Type	A complex type element that specifies the method and address(es) used for dynamic translation. Contains either an interface role object or address pool. If the interface role policy object is referenced then the globally registered IP address assigned to a particular interface will be used as the translated address.
translatedAddress.interfaceGID	Object identifier	An ObjectIdentifier ID that references an InterfaceRole Policy Object.
translatedAddress.addressPool	String	One or more address ranges, including the prefix, using the format min1-max1/prefix (in CIDR notation), where "prefix" represents a valid netmask. For example, `172.16.0.0-172.31.0.223/12`.
Settings	Complex Type	A complex type element that includes optional settings.
settings.enablePortTrans	Boolean	If true, the router uses port addressing (PAT) if supply of global addresses in the address pool is depleted; when false, PAT is not used.
settings.noTransVPN	Boolean	If true, address translation is not performed on VPN traffic. When false, the router performs address translation on VPN traffic in cases of overlapping addresses between the NAT ACL and the crypto ACL.

Table 40: InterfaceNATDynamicRulesRouterPolicy Class Definition

```
<xs:complexType name="InterfaceNATDynamicRulesRouterPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="trafficFlowAclObjectGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="translated" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:choice>
                                <xs:element name="interfaceGID" type="ObjectIdentifier"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="addressPool" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="settings" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="enablePortTrans" type="xs:boolean"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="noTransVPN" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 42: InterfaceNATDynamicRulesRouterPolicy XML Schema

3.1.5.10 DeviceNATTimeoutsRouterPolicy

A DeviceNATTimeoutsRouterPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a DeviceNATTimeoutsRouterPolicy specifies NAT timeout values for port address (overload) translations.

The following table defines the contents of a DeviceNATTimeoutsRouterPolicy:

Element. Sub Element	Type	Comment
maxEntries	unsignedLong	The maximum number of entries allowed in the dynamic NAT table. Corresponds to a value between 1 and 2147483647. If not specified, it means that the number of entries in the table is unlimited.
timeoutInSecs	unsignedLong	The number of seconds after which dynamic translations expire; this does not apply to PAT (overload) translations. The default is 86400 seconds (24 hours).
udpTimeoutInSecs	unsignedLong	The timeout value applied to User Datagram Protocol (UDP) ports. The default is 300 seconds (5 minutes).
dnsTimeoutInSecs	unsignedLong	The timeout value applied to Domain Naming System (DNS) server connections. The default is 60 seconds.
tcpTimeoutInSecs	unsignedLong	The timeout value applied to Transmission Control Protocol (TCP) ports. The default is 86400 seconds (24 hours).
finRstTimeoutInSecs	unsignedLong	The timeout value applied when a Finish (FIN) packet or Reset (RST) packet (both of which terminate connections) is found in the TCP stream. The default is 60 seconds.
icmpTimeoutInSecs	unsignedLong	The timeout value applied to Internet Control Message Protocol (ICMP) flows. The default is 60 seconds.
pptpTimeoutInSecs	unsignedLong	The timeout value applied to NAT Point-to-Point Tunneling Protocol (PPTP) flows. The default is 86400 seconds (24 hours).
synTimeoutInSecs	unsignedLong	The timeout value applied to TCP flows after a synchronous transmission (SYN) message (used for precise clocking) is encountered. The default is 60 seconds.

Table 41: DeviceNATTimeoutsRouterPolicy Class Definition

```
    <xs:complexType name="DeviceNATTimeoutsRouterPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="maxEntriesInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="timeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="udpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="dnsTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="tcpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="finRstTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="icmpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="pptpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="synTimeoutInSecs" type="xs:unsignedLong" minOccurs="0"
maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
```

Figure 43: DeviceNATTimeoutsRouterPolicy

3.1.5.11 InterfaceNATAddressPoolFirewallPolicy

An InterfaceNATAddressPoolFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of an InterfaceNATAddressPoolFirewallPolicy manages the global address pools used in dynamic NAT rules. This policy is applicable for PIX, FWSM and pre-ASA 8.3.

The following table defines the contents of an InterfaceNATAddressPoolFirewallPolicy:

Element. Sub Element	Type	Comment
interfaceGID	ObjectIdentifier	References a InterfaceRole Policy object GID interface on which the mapped IP addresses will be used.
poolId	unisignedInt	A unique identification number for this address pool, an integer between 1 and 2147483647. When configuring a dynamic NAT rule, the Pool ID is used to specify the pool of addresses to be used for translation
ipAddressRange	Complex Type	A complex type element containing the address(es) to be assigned to this address pool. The address can contain a combination of literal IPv4 addresses and/or reference to network policy objects.
ipAddressRange.ipv4Data	String	A literal IP Address.
ipAddressRange. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
interfaceKeyword	String	If the “interface” keyword is specified it means port address translation is enabled on the specified interface.

Table 42: InterfaceNATAddressPoolFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNATAddressPoolFirewallPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence>
                <xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="poolId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                <xs:element name="ipAddressRange" type="NetworkObjectsRefs" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="interfaceKeyword" type="xs:string" fixed="interface"
minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 44: InterfaceNATAddressPoolFirewallPolicy XML Schema

3.1.5.12 DeviceNATTransOptionsFirewallPolicy

A DeviceNATTransOptionsFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a DeviceNATTransOptionsFirewallPolicy manages the options that affect network address translation for the selected security appliance. These settings apply to all interfaces on the device.

This policy is applicable for PIX, FWSM and ASA.

The following table defines the contents of a DeviceNATTransOptionsFirewallPolicy:

Element. Sub Element	Type	Comment
isEnableTrafficWithoutTrans	boolean	It true, lets traffic pass through the security appliance without address translation. If this option is false, any traffic that does not match a translation rule will be dropped. **Note** This option is available only on PIX 7.x, FWSM 3.x, and ASA devices.
isXlateByPass	boolean	If true, NAT session for imtranslated traffic are disabled. **Note** This option is available only on FWSM 3.2 and higher

Table 43: DeviceNATTransOptionsFirewallPolicy Class Definition

```
<xs:complexType name="DeviceNATTransOptionsFirewallPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence>
                <xs:element name="isEnableTrafficWithoutTrans" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="isXlateByPass" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
```

Figure 45: DeviceNATTransOptionsFirewallPolicy XML Schema

3.1.5.13 InterfaceNATTransExemptionsFirewallPolicy

An InterfaceNATTransExemptionsFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of an InterfaceNATTransExemptionsFirewallPolicy specifies rules that exempt traffic from address translation. Rules are evaluated sequentially in the order listed.

This policy is applicable for PIX, FWSM and pre-ASA 8.3.

The following table defines the contents of an InterfaceNATTransExemptionsFirewallPolicy:

Element. Sub Element	Type	Comment
isRuleEnabled	boolean	If true, the rule is enabled and false indicates that the rule is disabled.
isExempt	boolean	If true, the rule identifies traffic that is exempt from NAT. If false, the rule identifies traffic that is not exempt from NAT.
realInterfaceGID	ObjectIdentifier	The device interface to which the rule is applied.
original	Complex Type	Complex type containing IP addresses for the source hosts and network objects to which the rule applies. Can contain multiple literal IP addresses and/or reference to network policy objects
original.ipv4Data	String	A literal IP Address.
original. interfaceRoleObjectGIDs	Object Identifier	List of Interface role Policy Object GIDs.
original. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
outsideNAT	boolean	True indicates rule outside keyword is defined on the NAT rule.
destinations	Complex Type	Complex type containing IP addresses for the destination hosts and network objects to which the rule applies. Can contain multiple literal IP addresses and/or reference to network policy objects
destinations.ipv4Data	String	A literal IP Address.
destinations. interfaceRoleObjectGIDs	Object Identifier	List of Interface role Policy Object GIDs.
destinations. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
fwsmAdvancedOptions	Complex Type	Advanced options applicable only for *FWSM*
fwsmAdvancedOptions.isTransDNSReplies	boolean	If true, the security appliance rewrites DNS replies so an outside client can resolve the name of an inside host using an inside DNS server, and vice versa.
fwsmAdvancedOptions.maxTCPConnPerRule	UnsignedInt	The maximum number of TCP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
fwsmAdvancedOptions.maxUDPConnPerRule	UnsignedInt	The maximum number of UDP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
fwsmAdvancedOptions.maxEmbConnections	UnsignedInt	The maximum number of embryonic connections allowed to form before the security appliance begins to deny these

Element. Sub Element	Type	Comment
		connections. Valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
fwsmAdvancedOptions.randomizeSeqNum	boolean	If true, the security appliance randomizes the sequence numbers of TCP packets

Table 44: InterfaceNATTransExemptionsFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNATTransExemptionsFirewallPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="isExempt" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="original" type="NetworkObjectsRefs" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="outsideNAT" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="destinations" type="NetworkObjectsRefs" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="fwsmAdvancedOptions" type="FirewallNATAdvancedOptions"
minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
```

Figure 46: InterfaceNATTransExemptionsFirewallPolicy XML Schema

3.1.5.14 InterfaceNATDynamicRulesFirewallPolicy

An InterfaceNATDynamicRulesFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a InterfaceNATDynamicRulesFirewallPolicyspecifies dynamic NAT and PAT rules. Rules are evaluated sequentially in the order listed.

This policy is applicable for PIX, FWSM and pre-ASA 8.3.

The following table defines the contents of an InterfaceNATDynamicRulesFirewallPolicy:

Element. Sub Element	Type	Comment
isRuleEnabled	boolean	If true, the rule is enabled and false indicates that the rule is disabled.
realInterfaceGID	ObjectIdentifier	Maps to the the device interface role policy object to which the rule applies.
poolId	Unsigned Int	The ID number of the pool of addresses used for translation. A value of zero to specify this as an identity NAT rule.
original	Complex Type	Complex type containing IP addresses for the source hosts and network objects to which the rule applies. Can contain multiple literal IP addresses and/or reference to network policy objects
original.ipv4Data	String	A literal IP Address.
original. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
outsideNAT	Boolean	If true, indicates the “outside” keyword is present on this NAT rule.
advancedOptions	Complex Type	Advanced options.
advancedOptions.isTransDNSReplies	boolean	If true, the security appliance rewrites DNS replies so an outside client can resolve the name of an inside host using an inside DNS server, and vice versa.
advancedOptions.maxTCPConnPerRule	UnsignedInt	The maximum number of TCP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.maxUDPConnPerRule	UnsignedInt	The maximum number of UDP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.maxEmbConnections	UnsignedInt	The maximum number of embryonic connections allowed to form before the security appliance begins to deny these connections. Valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.randomizeSeqNum	boolean	If true, the security appliance randomizes the sequence numbers of TCP packets

Table 45: InterfaceNATDynamicRulesFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNATDynamicRulesFirewallPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence>
                <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="poolId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                <xs:element name="original" type="NetworkObjectsRefs" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="outsideNAT" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                <xs:element name="advancedOptions" type="FirewallNATAdvancedOptions"
minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 47: InterfaceNATDynamicRulesFirewallPolicy XML Schema

3.1.5.15 InterfaceNATPolicyDynamicRulesFirewallPolicy

An InterfaceNATPolicyDynamicRulesFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a InterfaceNATPolicyDynamicRulesFirewallPolicy specifies dynamic translation rules based on source and destination addresses and services. Rules are evaluated sequentially in the order listed.

This policy is applicable for PIX, FWSM and pre-ASA 8.3.

The following table defines the contents of an InterfaceNATPolicyDynamicRulesFirewallPolicy:

Element. Sub Element	Type	Comment
isRuleEnabled	boolean	If true, the rule is enabled and false indicates that the rule is disabled.
realInterfaceGID	ObjectIdentifier	Maps to the the device interface role policy object to which the rule applies.
poolId	Unsigned Int	The ID number of the pool of addresses used for translation. A value of zero to specify this as an identity NAT rule.
Original	Complex Type	Complex type containing IP addresses for the source hosts and network objects to which the rule applies. Can contain multiple literal IP addresses and/or reference to network/interface role policy objects
original.ipv4Data	String	A literal IP Address.
original. interfaceRoleObjectGIDs	ObjectIdentifierList	List of interface role policy objects.
original. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
outsideNAT	Boolean	If true, indicates the “outside” keyword is present on this NAT rule.
destinations	Complex Type	Complex type containing IP addresses for the destination hosts and network objects to which the rule applies. Can contain multiple literal IP addresses and/or reference to network/interface role policy objects
destinations.ipv4Data	String	A literal IP address.
destinations. interfaceRoleObjectGIDs	ObjectIdentifierList	List of interface role policy objects.
destinations. networkObjectGID	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
Services	Complex Type	Complex type containing services to which the rule applies. This can be a combination of service information in the format protocol/source-port/destination-port and/or references to Service Policy object
services.serviceData	String	The syntax for service specification is: **{tcp | udp | tcp&udp}/{source_port_number | port_list_object}/ {destination_port_number | port_list_object}**

Element. Sub Element	Type	Comment
		If only one port parameter, it refers tp the destination port (with a source port of "any"). For example, **tcp/4443** means tcp, source port any, destination port 4443, while **tcp/4443/Default Range** means tcp, source port 4443, and destination port Default Range (generally 1-65535).
services. serviceObjectGID	Object identifier	An ObjectIdentifier ID that references a Service Policy Object.
advancedOptions	Complex Type	Advanced options.
advancedOptions.isTransDNSReplies	boolean	If true, the security appliance rewrites DNS replies so an outside client can resolve the name of an inside host using an inside DNS server, and vice versa.
advancedOptions.maxTCPConnPerRule	UnsignedInt	The maximum number of TCP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.maxUDPConnPerRule	UnsignedInt	The maximum number of UDP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.maxEmbConnections	UnsignedInt	The maximum number of embryonic connections allowed to form before the security appliance begins to deny these connections. Valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.randomizeSeqNum	boolean	If true, the security appliance randomizes the sequence numbers of TCP packets

Table 46: InterfaceNATPolicyDynamicRulesFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNATPolicyDynamicRulesFirewallPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="poolId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="original" type="NetworkInterfaceObjectsRefs" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="outsideNAT" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="destinations" type="NetworkInterfaceObjectsRefs" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="services" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="serviceData" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                                <xs:element name="serviceObjectGID" type="ObjectIdentifier"
minOccurs="0" maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="advancedOptions" type="FirewallNATAdvancedOptions"
minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 48: InterfaceNATPolicyDynamicRulesFirewallPolicy XML Schema

3.1.5.16 InterfaceNATStaticRulesFirewallPolicy

An InterfaceNATStaticRulesFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of an InterfaceNATStaticRulesFirewallPolicy specifies static translation rules for a security appliance. Rules are evaluated sequentially in the order listed.

This policy is applicable for PIX, FWSM and pre-ASA 8.3.

The following table defines the contents of an InterfaceNATStaticRulesFirewallPolicy:

Element. Sub Element	Type	Comment
isRuleEnabled	boolean	If true, the rule is enabled and false indicates that the rule is disabled.
translationType	Enumeration	Type of translation for the rule - “NAT” or “PAT”.
realInterfaceGID	ObjectIdentifier	Maps to the the device interface role policy object to which the rule applies.
mappedInterfaceGID	Object identifier	Maps to the interface role policy object interface on which the translated addresses are to be used.
original	Complex Type	Complex type containing IP addresses for the source hosts and network objects to which the rule applies. Can contain multiple literal IP addresses and/or reference to network policy objects
original.ipv4Data	String	A literal IP Address.
original. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
Translated	Complex Type	A complex type element containing the translated addresses.
translated.ipv4Data	String	A literal IP address.
translated. networkObjectGID	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
translated.interfaceKeyword	String	A value of “interface” specifies that this keyword is present in the NAT rule
policyNAT	Complex Type	Complex Type containing Policy NAT details only if Policy NAT is enabled for this rule
policyNAT.destAddress	Complex Type	Complex Type containing the destination addresses.
policyNAT.destAddress.ipv 4Data	String	A literal IP Address.
policyNAT.destAddress. networkObjectGIDs	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
policyNAT.services	Complex Type	Complex Type that specifies the services to which the rule applies. This can be a combination of service information in the format protocol/source-port/destination-port and/or references to Service Policy object
policyNAT.services.service Data	String	The syntax for service specification is: **{tcp | udp | tcp&udp}/{source_port_number | port_list_object}/ {destination_port_number |**

Element. Sub Element	Type	Comment
		port_list_object} If only one port parameter, it refers tp the destination port (with a source port of "any"). For example, **tcp/4443** means tcp, source port any, destination port 4443, while **tcp/4443/Default Range** means tcp, source port 4443, and destination port Default Range (generally 1-65535).
policyNAT.services. serviceObjectGID	Object identifier	An ObjectIdentifier ID that references a Service Policy Object.
Protocol	Enumeration	Protocol – "UDP" , "TCP", "IP" to which the rule applies.
originalPort	Unsigned Int	If PAT is the selected Translation Type, this specifies the port number to be translated
translatedPort	Unsigned Int	If PAT is the selected Translation Type, this specifies the port number to which the original port number will be translated.
advancedOptions	Complex Type	Advanced options.
advancedOptions.isTransDNSReplies	boolean	If true, the security appliance rewrites DNS replies so an outside client can resolve the name of an inside host using an inside DNS server, and vice versa.
advancedOptions.maxTCPConnPerRule	UnsignedInt	The maximum number of TCP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.maxUDPConnPerRule	UnsignedInt	The maximum number of UDP connections allowed; valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.maxEmbConnections	UnsignedInt	The maximum number of embryonic connections allowed to form before the security appliance begins to deny these connections. Valid values are 0 through 65,535. If this value is set to zero, the number of connections is unlimited.
advancedOptions.randomizeSeqNum	boolean	If true, the security appliance randomizes the sequence numbers of TCP packets

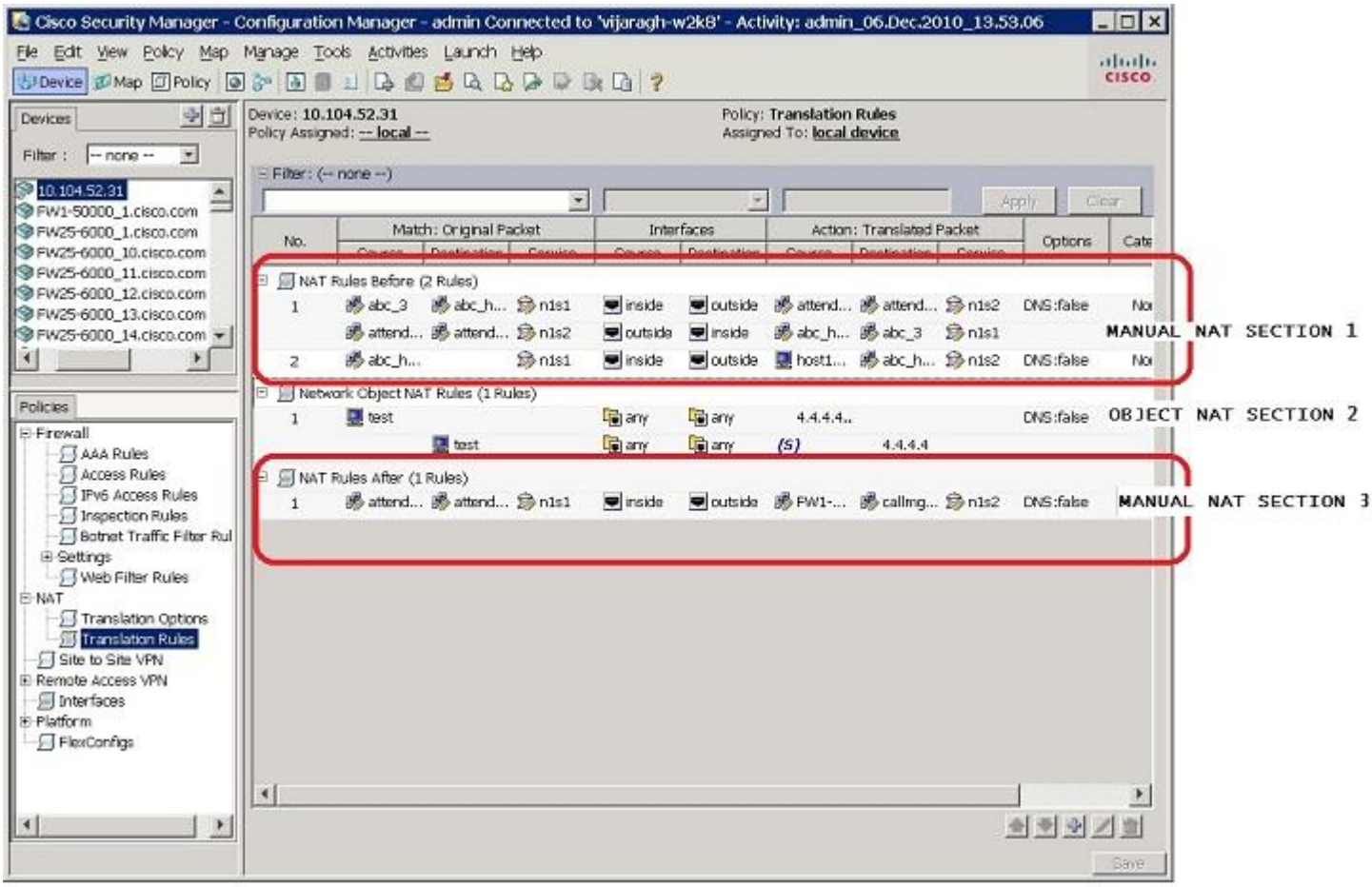
Table 47: InterfaceNATStaticRulesFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNATStaticRulesFirewallPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence>
                <xs:element name="isRuleEnabled" type=xs:boolean" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="translationType" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="NAT"/>
                            <xs:enumeration value="PAT"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="mappedInterfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="original" type="NetworkOrIPRef" minOccurs="1" maxOccurs="1"/>
                <xs:element name="translated" type="NetworkObjectRefs" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="policyNAT" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="destAddress" type="NetworkObjectsRefs"
minOccurs="1" maxOccurs="1"/>
                            <xs:element name="services" minOccurs="1" maxOccurs="1">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="serviceData" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element name="serviceObjectGID"
type="ObjectIdentifier" minOccurs="0" maxOccurs="unbounded"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="protocol" type="IPTransportProtocol" minOccurs="1"
maxOccurs="1"/>
                <xs:element name="originalPort" type="xs:unsignedInt" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="translatedPort" type="xs:unsignedInt" minOccurs="0"
maxOccurs="1"/>
                <xs:element name="advancedOptions" type="FirewallNATAdvancedOptions"
minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
```

Figure 49: InterfaceNATStaticRulesFirewallPolicy XML Schema

3.1.5.17 InterfaceNATManualFirewallPolicy

This policy is only applicable for device **ASA Version 8.3 or later**. Beginning with ASA 8.3, the complete NAT representation is a combination of two key policy types. This consists of Manual NAT pre and post rules (section 1 and 3) and object NAT rules (section 2) as shown below.



This section covers the Manual NAT (InterfaceNATManualFirewallPolicy) and the following section covers the Object NAT (InterfaceNATObjectFirewallPolicy).

An InterfaceNATManualFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a InterfaceNATManualFirewallPolicy specifies the "manually" defined NAT rules on the device. The base **orderId** attribute that is inherited from the base policy specifies the ordering of these rules in the policy. The **section** element in the Policy data specifies whether this is a pre or post NAT rule.

The following table defines the contents of an InterfaceNATManualFirewallPolicy:

Element. Sub Element	Type	Comment
isRuleEnabled	boolean	If true, the rule is enabled and false indicates that the rule is disabled.
section	Enumeration	Specifies the rule section. Valid values are "1", "2" and "3".

Element. Sub Element	Type	Comment
		Following is the interpretation "1" → Indicates pre-NAT or NAT before rules "2" → Indicates Object NAT rules "3" → Indicates post-NAT or NAT rules after. For this policy type only "1" and "3" are possible. The BasePolicy orderId element will specify the "ordering" of the rules within this section.
realInterface	Complex Type	Reference to an interface role Policy Object or interface name
realInterface.realInterfaceGID	Object Identifier	Reference to an interface role Policy Object
realInterface.realInterfaceName	String	The real interface name
mappedInterface	Complex Type	Reference to an interface role Policy Object or name (Note: if both the real and mapped interfaces are not specified then the default to be assumed as "any")
mappedInterface.mappedInterfaceGID	Object Identifier	Reference to an interface role Policy Object
mappedInterface.mappedInterfaceName	String	The mapped interface name.
Source	Complex Type	Complex Type containing the Original and translated sources
source.natType	Enumeration	Specifies the type of translation rule either "Static" or "Dynamic".
source.originalObjectGID	Object Identifier	Specifies the source address the NAT rule will translate. If this is a range or network, all addresses in the range or network are translated. This element includes an ObjectIdentifier ID that references a Network Policy Object.
source.translated	Complex Type	Complex Type containing **either** a pool of network address or reference to an interface object representation the translated source.
source.translated.ObjectGID	Object Identifier	Reference to a network Policy Object GID. **Literal IPv4 addresses are not allowed for Manual NAT configurations**.
source.translated.interfaceKeyword	String	The value of "interface" specifies that the interface keyord has been applied for the NAT rule.
source.translated.patPool	Complex Type	Complex Type containing the PAT options
source.translated.patPool.patAddressPool	Complex Type	Complex Type containing the PAT address pool
source.translated.patPool.patAddressPool.patPoolAddressGID	Object Identifier	Reference to a network policy object

Element. Sub Element	Type	Comment
source.translated.patPool. patAddressPool.. interfaceKeyword	String	Interface to provide a Fallthrough Interface, it will be same as destination interface if the fixed string “interface” is used as the value for this element.
source.translated.patPool. isPatAllocatedInRoundRobin	boolean	If true, it means a "round robin" cycling through available IP addresses and port numbers. This method assigns an address/port combination using each successive address in the pool; it then uses the first address again with a different port, proceeds to the second address again, and so on
Destination	Complex Type	Complex Type containing the the static translation of destination addresses
destination.natType	Enumeration	Specifies the type of translation rule either “Static” or “Dynamic”.
destination.originalObject	Complex Type	Reference to the original object
destination. originalObject.networkObjectGIDs	Object IdentifierList	Reference to a list of network Policy Object GIDs.
destination. originalObject.ipV4Data	String	Destination IPV4 address.
destination.translatedObjectGID	ObjectIdentifier	Reference to the translated network object GID.
Service	Complex Type	Complex Type that specifies port address translation.
service.originalObjectGID	Object Identifier	Reference to a Service Policy object that defines the service(s) to be translated.
service.transObjectGID	Object Identifier	Reference to a Service Policy object that defines the service(s) to be used for translation.
isTransDNSReplies	boolean	If true, addresses embedded in DNS replies that match this rule are rewritten.
Direction	Enumeration	Specifies whether a static NAT rule in a single direction only (“Unidirectional”); or dual rules (“Bidirectional”), one each for both directions (forward and reverse).
isNoProxyARP	boolean	If true, this disables proxy ARP on the specified Destination Interface. By default, all NAT rules include proxy ARP on the egress interface. A NAT Exempt rule is used to bypass NAT for both ingress and egress traffic, relying on route look-up to locate the egress interface. Thus, Proxy ARP should be disabled for NAT Exempt rules. (The NAT Exempt rules always take priority and appear above all other NAT rules in the Translation Rules table.)
isRouteLookUp	boolean	If this option is true, the egress interface is determined using route look-up instead of using the specified Destination Interface. This must be true for a NAT Exempt rule.

Table 48: InterfaceNATManualFirewallPolicy Class Definition

```
<xs:complexType name="PatOptions">
            <xs:sequence>
                  <xs:element name="patAddressPool" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                              <xs:choice>
                                    <xs:element name="patPoolAddressGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1"/>
                                    <xs:element name="interfaceKeyword" type="xs:string" fixed="interface" minOccurs="0"
maxOccurs="1"/>
                              </xs:choice>
                        </xs:complexType>
                  </xs:element>
                  <xs:element name="isPatAllocatedInRoundRobin" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
       </xs:sequence>
      </xs:complexType>
      <xs:complexType name="InterfaceNATManualFirewallPolicy">
            <xs:complexContent>
                  <xs:extension base="BasePolicy">
                        <xs:sequence>
                              <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="section" minOccurs="1" maxOccurs="1">
                                    <xs:simpleType>
                                          <xs:restriction base="xs:unsignedInt">
                                                <xs:enumeration value="1"/>
                                                <xs:enumeration value="2"/>
                                                <xs:enumeration value="3"/>
                                          </xs:restriction>
                                    </xs:simpleType>
                              </xs:element>
                              <xs:element name="realInterface" minOccurs="0" maxOccurs="1">
                                    <xs:complexType>
                                          <xs:choice>
                                                <xs:element name="realInterfaceGID" type="ObjectIdentifier"
minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="realInterfaceName" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                          </xs:choice>
                                    </xs:complexType>
                              </xs:element>
                              <xs:element name="mappedInterface" minOccurs="0" maxOccurs="1">
                                    <xs:complexType>
                                          <xs:choice>
                                                <xs:element name="mappedInterfaceGID" type="ObjectIdentifier"
minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="mappedInterfaceName" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                          </xs:choice>
                                    </xs:complexType>
                              </xs:element>
                              <xs:element name="source" minOccurs="1" maxOccurs="1">
                                    <xs:complexType>
                                          <xs:sequence>
                                                <xs:element name="natType" type="NATType" minOccurs="1"
maxOccurs="1"/>
                                                <xs:element name="originalObjectGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                                                <xs:element name="translated" minOccurs="0" maxOccurs="1">
                                                      <xs:complexType>
```

Figure 50: InterfaceNATManualFirewallPolicy

3.1.5.18 InterfaceNAT64ManualFirewallPolicy

An InterfaceNAT64ManualFirewallPolicy represents a Unified (IPv6/IPv4) Manual NAT Rule. The Unified NAT Rules are supported from ASA 9.0 onwards. The network translations supported are

- IPv4 -> IPv6
- IPv6 -> IPv4
- IPv6 -> IPv6
- IPv4 -> IPv4

InterfaceNAT64ManualFirewallPolicy extends from the base InterfaceNATManualFirewallPolicy class An instance of a InterfaceNAT64ManualFirewallPolicy denotes a single Unified NAT Rule.

This policy is available beginning with Version 1.1 of the API.

Element	Type	Comment
isInterfaceIPv6	boolean	Use IPv6 Address of the interface
isNetToNet	boolean	Option for one-to-one mapping of single IPv6 address to an IPv4 server

Table 49: InterfaceNAT64ManualFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNAT64ManualFirewallPolicy">
   <xs:complexContent>
      <xs:extension base=" InterfaceNATManualFirewallPolicy ">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="isInterfaceIPv6" type="boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isNetToNet" type="boolean" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
      </xs:extension>
</xs:complexContent>
</xs:complexType>
```

Figure 51: InterfaceNAT64ManualFirewallPolicy XML Schema

3.1.5.19 InterfaceNATObjectFirewallPolicy

An InterfaceNATObjectFirewallPolicy extends from the base BasePolicy class and inherits all its attributes. An instance of a InterfaceNATObjectFirewallPolicy specifies the object NAT rules on the device. The base **order-id** attribute that is inherited from the base policy specifies the ordering of these rules in the policy.

This policy is applicable only for devices running **ASA Version 8.3 or later.**

The following table defines the contents of an InterfaceNATObjectFirewallPolicy:

Element. Sub Element	Type	Comment
section	Enumeration	Specifies the rule section. Valid values are "1", "2" and "3". Following is the interpretation "1" → Indicates pre-NAT or NAT before rules "2" → Indicates Object NAT rules "3" → Indicates post-NAT or NAT rules after. For this policy type only "2" is allowed. The BasePolicy orderId element will specify the "ordering" of the rules within this section.
realInterface	String	Interface string
mappedInterface	String	Interface String
natType	Enumeration	Specifies the type of translation rule either "Static" or "Dynamic".
originalObjectGID	String	The source address the NAT rule will translate.
Translated	Complex Type	Complex type that specifies whether the translation is based on **either** an address or interface.
translated.objectGID	Complex Type	Complex Type containing the address definitions
translated.objectGID.ipv4Data	String	A literal IP address.
translated.objectGID.networkObjectGID	Object identifier	An ObjectIdentifier ID that references a Network Policy Object.
translated.objectGID.interfaceKeyword	String	A value of "interface" indicates that the interface keyword is defined for this NAT rule.
translated.patPool	Complex Type	On ASA Version 8.4.2 and later, a separate PAT Pool for a Dynamic NAT and PAT rule can be defined. The PAT Pool addresses are specified using this PAT Pool Address Translation field. This contains the PAT Pool options.
translated.patPool.patAddressPool	Complex Type	Containing the address information
translated.patPool.patAddressPool.patPoolAddressGID	Object identifier	An ObjectIdentifier ID that references a Network Policy Object for PAT Pool.
translated.patPool.patAddressPool.interfaceKey	String	A value of "interface" indicates that the interface keyword is

Element. Sub Element	Type	Comment
word		defined for this NAT rule.
translated.patPool . isPatAllocatedInRound Robin	boolean	If this is true ASA Device uses Round Robin Allocation for PAT Pool
isTransDNSReplies	boolean	If true, addresses embedded in DNS replies that match this rule are rewritten.
isNoProxyARP	boolean	If true, do not proxy ARP on Destination Interface
isRouteLookUp	boolean	If true, perform Route Lookup for Destination Interface
Service	Complex Type	Complex Type that defines configuration static port address translation. Application only for static rules
service.protocol	Enumeration	Protocol – “UDP”, “TCP” or “IP” to which the rule applies.
service.originalPort	Unsigned Int	The port on which the traffic enters the device.
service.transPort	Unsigned Int	The port number which is to replace the original port number.

Table 50: InterfaceNATObjectFirewallPolicy Class Definition

```
<xs:complexType name="InterfaceNATObjectFirewallPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence>
                <xs:element name="section" fixed="2" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:unsignedInt">
                            <xs:enumeration value="1"/>
                            <xs:enumeration value="2"/>
                            <xs:enumeration value="3"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="realInterface" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="mappedInterface" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="natType" type="NATType" minOccurs="1" maxOccurs="1"/>
                <xs:element name="originalObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                 <xs:element name="translated" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="objectGID" type="NetworkObjectRefs" minOccurs="0"
maxOccurs="1"/>
                            <xs:element name="patPool" type="PatOptions" minOccurs="0"
maxOccurs="1"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="isTransDNSReplies" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                <xs:element name="isNoProxyARP" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="isRouteLookUp" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                <xs:element name="service" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="protocol" type="IPTransportProtocol" minOccurs="1"
maxOccurs="1"/>
                            <xs:element name="originalPort" type="xs:unsignedInt" minOccurs="0"
maxOccurs="1"/>
                            <xs:element name="transPort" type="xs:unsignedInt" minOccurs="0"
maxOccurs="1"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
```

Figure 52: InterfaceNATObjectFirewallPolicy XML Schema

3.1.5.20 InterfaceNAT64ObjectFirewallPolicy

An InterfaceNAT64ObjectFirewallPolicy represents a Unified (IPv6/IPv4) Object NAT Rule. The Unified NAT Rules are supported from ASA 9.0 onwards. The network translations supported are

- IPv4 -> IPv6
- IPv6 -> IPv4
- IPv6 -> IPv6
- IPv4 -> IPv4

InterfaceNAT64ObjectFirewallPolicy extends from the base InterfaceNATObjectFirewallPolicy class. An instance of a InterfaceNAT64ObjectFirewallPolicy denotes a single Unified Object NAT Rule.

This policy is available beginning with Version 1.1 of the API.

Element	Type	Comment
isInterfaceIPv6	boolean	Use IPv6 Address of the interface
isNetToNet	boolean	Option for one-to-one mapping of single IPv6 address to an IPv4 server

Table 51: InterfaceNAT64ObjectFirewallPolicy Class Definition

```
<xs:complexType name=" InterfaceNAT64ObjectFirewallPolicy ">
   <xs:complexContent>
      <xs:extension base=" InterfaceNATObjectFirewallPolicy ">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="isInterfaceIPv6" type="boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isNetToNet" type="boolean" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
      </xs:extension>
</xs:complexContent>
</xs:complexType>
```

Figure 53: InterfaceNAT64ObjectFirewallPolicy XML Schema

3.2 Methods

The configuration service defines the following methods:

1. GetServiceInfo

    a. Returns the service-specific information, including name of service, version, date, etc.
    b. This method should be called after authentication has returned a valid session cookie and prior to any other method to ensure the client is compatible with the service version currently running.

2. GetGroupList
    a. Returns the list of device groups defined in the system
3. GetDeviceListByCapability
    a. Returns the list of all devices that match one or more of the capabilities or wildcard in the system.
4. GetDeviceListByGroup
    a. Returns the list of all devices within a named group in the system.
5. GetDeviceConfigByGID
    a. Returns the device configuration identified by the object identifier specified in the method arguments.
6. GetDeviceConfigByName
    a. Returns the device configuration identified by the device's name specified in the method arguments.
7. GetPolicyConfigByName
    a. Returns a policy configuration that is assigned to more than one device that has been assigned a name.
8. GetPolicyConfigByDeviceGID
    a. Returns the policy configuration belonging to the device's object identifier that matches the particular type specified in the method arguments
9. GetPolicyListByDeviceGID
    a. Returns the list of policy names and their types belonging to the device's object identifier specified in the method arguments
10. GetSharedPolicyListByType
    a. Returns a list of all the shared Policies defined in CSM for a given policy type.

3.2.1 Method GetServiceInfo

The GetServiceInfo method returns the service description, information, and pertinent attributes related to the service.

3.2.1.1 Request

An example of the method GetServiceInfo request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/GetServiceInfo

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
   <getServiceInfoRequest>
          <protVersion>1.0</protVersion>
          <reqId>123</reqId>
   </getServiceInfoRequest>
```

Figure 54: Method GetServiceInfo Request Example

Table 52: Method GetServiceInfo Request URL Argument Descriptions

HTTP/XML Content	Definition
getServiceInfoRequest	Object that contains request arguments
HTTP Method	PUT
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="getServiceInfoRequest" type="GetServiceInfoRequest"/>
<xs:complexType name="GetServiceInfoRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
      </xs:complexContent>
</xs:complexType>
```

Figure 55: GetServiceRequest XML Schema

3.2.1.2 Response

An example of the GetServiceInfo response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ns1:getServiceInfoResponse xmlns:ns1="csm">
   <protVersion>1.0</protVersion>
   <reqId>123</reqId>
   <serviceVersion>1.0</serviceVersion>
   <serviceName>CSM Configuration Service</serviceName>
   <serviceDesc>A configuration service that enables network services configuration to be
   retrieved</serviceDesc>
</ns1:getServiceInfoResponse>
```

Figure 56: GetServiceInfo Response Example

Table 53: GetServiceInfo Response Elements and Attributes Description

XML Element & Attributes	Definition
getServiceInfoResponse	Returns the service information of the configuration service
serviceVersion	The service version of the configuration service running
serviceName	The service name
serviceDescr	The service description that provides a more detailed explanation of the service capabilities and features

```
<xs:element name="getServiceInfoResponse" type="GetServiceInfoResponse"/>
<xs:complexType name="GetServiceInfoResponse">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="serviceVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="serviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="serviceDesc" type="xs:string" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 57: GetServiceInfoResponse XML Schema

3.2.2 Method GetGroupList

The GetGroupList method returns the list of devices matching a particular type or all devices if the wildcard argument is chosen.

3.2.2.1 Request

An example of the method GetGroupList request is shown in the figure below.. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getGroupList

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<groupListRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <includeEmptyGroups>false</includeEmptyGroups>
</groupListRequest>
```

Figure 58: Method GetGroupList Request Example

Table 54: Method GetGroupList Request URL Argument Descriptions

HTTP/XML Content	Definition
GroupListRequest	Group list request that contains request arguments
Element: includeEmptyGroups	An element that identifies whether the response should include groups that have no devices within them or not.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="groupListRequest" type="GroupListRequest"/>
<xs:complexType name="GroupListRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="includeEmptyGroups" type="xs:boolean"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 59: GroupListRequest XML Schema

3.2.2.2 Response

An example of the GroupListRequest response is shown in the figure below. The fields in these messages are described in the table below. In a GetGroupListResponse, the full config of the device *will not be set*. This is only set when the GetDeviceConfigByName or GetDeviceConfigByGID methods are called.

```
<?xml version="1.0" encoding="UTF-8"?>
<groupListResponse>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <deviceGroup>
          <gid>00000000-0000-0000-0000-000000000001</gid>
          <name>San Jose</name>
          <lastUpdateTime>2011-05-22T07:07:29.129Z</lastUpdateTime>
          <path>/VmsVirtualRoot/San Jose</path>
          <deviceGroup>
            <gid>00000000-0000-0000-0000-000000000002</gid>
            <name>Building 13</name>
            <lastUpdateTime>2011-05-22T07:07:29.129Z</lastUpdateTime>
            <path>/VmsVirtualRoot/San Jose/Building 13</path>
              <device>
                 <gid>00000000-0000-0000-0000-214748364932</gid>
                 <name>10.77.208.138</name>
                 <lastUpdateTime>2011-05-26T00:11:53Z</lastUpdateTime>
                 <osType>asa</osType>
                 <osVersion>8.4(1)</osVersion>
                 <imageName>disk0:/asa831-k8.bin</imageName>
                 <sysObjectID>1.3.6.1.4.1.9.1.670</sysObjectID>
                 <fullConfig/>
                 <mgmtInterface>
                    <type>Management</type>
                    <identifier>mgmt</identifier>
                    <ipInterface>
                       <ipAddress>10.77.208.138/255.255.255.0</ipAddress>
                    </ipInterface>
                 </mgmtInterface>
                 <interfaceList>
                    <interface>
                       <type>GigabitEthernet</type>
                       <identifier>outside</identifier>
                       <ipInterface>
                            <ipAddress>20.10.30.42/255.255.255.0</ipAddress>
                       </ipInterface>
                    </interface>
                    <interface>
                       <type>FastEthernet</type>
                       <identifier>FastEthernet5/1</identifier>
                       <ipInterface>
                            <ipAddress>20.10.30.45/255.255.255.0</ipAddress>
                       </ipInterface>
                    </interface>
                 </interfaceList>
              <configState>committed</configState>
            </device>
          </deviceGroup>
        </deviceGroup>
```

Figure 60: GetGroupList Response Example

Table 55: GetGroupList Response Elements and Attributes Description

XML Element & Attributes	Definition
groupListResponse	Returns a list of 0 or more device groups
Element List: DeviceGroup	A list of group elements

```
<xs:element name="groupListResponse" type="GroupListResponse"/>
<xs:complexType name="GroupListResponse">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="deviceGroup" type="DeviceGroup"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Figure 61: GetGroupList Response XML Schema

Method specific errors:

Code	Description
2000	This error will be returned if the API is unable to retrieve the list of authorized devices for the user who is logged in.
2001	This error will be returned if API is not able to get the available groups from CSM Server for further processing.
2002	This error will be returned if API is not able to retrieve devices from CSM Server.
2003	This error will be returned if API is not able to find the config state of the device.
2004	This error will be returned if API is not able to retrieve interface details for a device.
2005, 2006	These errors will be returned if the API encounters an internal error when processing device specific data.

Table 56: GetGroupList Method Error Codes

3.2.3 Method GetDeviceListByCapability

The GetDeviceListByCapability method returns the list of devices matching one or more categories or all devices if the wildcard argument is chosen.

3.2.3.1 Request

An example of the method GetDeviceListByType request is shown in the figure below.. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getDeviceListByType

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<deviceListByCapabilityRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <deviceCapability>firewall</deviceCapability>
</deviceListByCapabilityRequest>
```

Figure 62: Method GetDeviceListByCapability Request Example

Table 57: Method GetDeviceListByCapability Request URL Argument Descriptions

HTTP/XML Content	Definition
getDeviceListByCapabilityRequest	Device list request that contains request arguments
Element: deviceCapability	One or more instances of an element that identifies the capability of the device being requested. Allowed values include: • **firewall**: To return all ASA, PIX and FWSM devices. • **ids**: To return all IPS Devices • **router**: To return routers • **switch**: To return switches Wildcard may be specified as <csm:deviceCapability>*</csm:deviceCapability>
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deviceListByCapabilityRequest" type="DeviceListByCapabilityRequest"/>
      <xs:complexType name="DeviceListByCapabilityRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="deviceCapability" type="DeviceCapability" minOccurs="1"
maxOccurs="unbounded"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 63: DeviceListByCapabilityRequest XML Schema

3.2.3.2 Response

An example of the GetDeviceListByCapability response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version="1.0" encoding="UTF-8"?>
<deviceListResponse>
         <protVersion>1.0</protVersion>
         <reqId>123</reqId>
         <deviceId>
           <gid>00000000-0000-0000-0000-12211312321</gid>
           <deviceCapability>firewall</deviceCapability>
           <ipv4Address>12.1.1.1</ipv4Address>
         </deviceId>
</deviceListResponse>
```

Figure 64: GetDeviceListByCapability Response Example

Table 58: GetDeviceListByCapability Response Elements and Attributes Description

XML Element & Attributes	Definition
deviceListResponse	Returns a list of 0 or more devices that match the filter parameter passed in the method
Element List: DeviceId	A list of device ID elements
Attribute: Gid	The gid attribute of the device
Element: deviceCapability	One or more capabilities of the device (mandatory)
Element: deviceName	Name of the device (mandatory)
Element: ipv4Address	The IPv4 address of the device (optional)

```
<xs:element name="deviceListResponse" type="DeviceListResponse"/>
    <xs:complexType name="DeviceListResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="deviceId" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence minOccurs="1" maxOccurs="1">
                                <xs:element name="deviceCapability" type="DeviceCapability"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="deviceName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
                                <xs:element name="ipv4Address" type="xs:string" minOccurs="0"
maxOccurs="1"/>
                                <xs:element name="gid" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 65: DeviceListResponse XML Schema

The error codes documented for the GetGroupList method are also applicable for this method.

3.2.4 Method GetDeviceListByGroup

The GetDeviceListByGroup method returns the list of devices contained within a particular group or all devices if the wildcard argument is chosen. The group name path will be prepared by combining the entire path Items mentioned in the request body. The list of devices will be from the combination of the path Items. For example, the path Item is provided as *San Jose* alone and the CSM Server has sub groups under San Jose then the API will return all the devices matching group *San Jose* and other sub groups under it as well.

For example if the path items are *San Jose* and *Building 14* then the devices matching group path *"/San Jose/Building 14*" and its sub group (if any) will be returned.

3.2.4.1 Request

An example of the method GetDeviceListByGroup request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getDeviceListByGroup

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<deviceListByGroupRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <deviceGroupPath>
          <pathItem>building10</pathItem>
          <pathItem>4th floor</pathItem>
        </deviceGroupPath>
</deviceListByGroupRequest>
```

Figure 66: Method GetDeviceListByGroup Request Example

Table 59: Method GetDeviceListByGroup Request URL Argument Descriptions

HTTP/XML Content	Definition
getDeviceListByGroupRequest	Device list request that returns a list of devices contained in the group
Element: deviceGroupPath	An element that identifies the device group being requested. Contains 1 or more *pathItem* entries. All the pathItems combined (separated by a slash "/") will be treated as one group name. The path information from the response of the GetGroupList method can be used here. Each group in the GetGroupList has path attribute which provide the complete path name. Each element in the path (separated by a "/") is encoded as pathItems. For example if device details for group includes a path such as "/VmsVirtualRoot/San Jose/Building 13" (obtained from the response of getGroupList API), then pathItems will be "VmsVirtualRoot", "San Jose" and "Building 13". The "VmsVirtualRoot" is a virtual "root node" for all groups.

HTTP/XML Content	Definition
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deviceListByGroupRequest" type="DeviceListByGroupRequest"/>
<xs:complexType name="DeviceListByGroupRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="deviceGroupPath" type="DeviceGroupPath"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 67: DeviceListByGroupRequest XML Schema

3.2.4.2 Response

The GetDeviceListByGroup response and error codes are the same as method GetDeviceListByCapability response.

3.2.5 Method GetDeviceConfigByGID

The GetDeviceConfigByGID method returns a specific device object and its associated configuration based on the device id passed into the method. The user requesting the device configuration must have both the "view_device" and "view_cli" privilege on the device for which the config is being requested. The data in the "fullConfig" element that this method returns is equal to the data from a typical "show running-config" command exectuted on the device. This full configuration is returned from CSM's Configuration Archive (CA) component. The CA is updated during device discovery and/or deployments. **NOTE**: *This fullConfig will not contain any Out Of Band (OOB) configuration updates done directly on the device without using CSM.*

Please also see Section 3.2.11 and Section 5.

3.2.5.1 Request

An example of the method GetDeviceConfigByGID request is shown in the figure below.. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getDeviceConfigByGID

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<deviceConfigByGIDRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <gid>00000000-0000-0000-0000-051539607555</gid>
</deviceConfigByGIDRequest>
```

Figure 68: Method GetDeviceConfigByGID Request Example

Table 60: Method GetDeviceConfigByGID Request URL Attribute Descriptions

URL Attribute Name	Definition
deviceConfigByGIDRequest	A global ID request
gid	The device global object identifier requested to be returned. **NOTE**: The device GID may be obtained by using any of the APIs - *getDeviceListByCapability* or *getDeviceListByGroup* or *GetGroupList*.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deviceConfigByGIDRequest" type="DeviceConfigByGIDRequest"/>
<xs:complexType name="DeviceConfigByGIDRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="gid" type="ObjectIdentifier"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 69: DeviceConfigByGIDRequest XML Schema

3.2.5.2 Response

An example of the GetDeviceConfigByGID response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version="1.0" encoding="UTF-8"?>
<deviceConfigResponse>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
            <device>
               <gid>00000000-0000-0000-0000-214748364932</gid>
               <name>10.77.208.138</name>
               <lastUpdateTime>2011-05-26T00:11:53Z</lastUpdateTime>
               <osType>asa</osType>
               <osVersion>8.4(1)</osVersion>
               <imageName>disk0:/asa831-k8.bin</imageName>
               <fullConfig>
                  <!— will contain the full config of the device -->
               </fullConfig>
               <mgmtInterface>
                  <type>Management</type>
                  <identifier>mgmt</identifier>
                  <ipInterface>
                     <ipAddress>10.77.208.138/255.255.255.0</ipAddress>
                  </ipInterface>
               </mgmtInterface>
               <interfaceList>
                  <interface>
                     <type>GigabitEthernet</type>
                     <identifier>outside</identifier>
                     <ipInterface>
                          <ipAddress>20.10.30.42/255.255.255.0</ipAddress>
                     </ipInterface>
                  </interface>
                  <interface>
                     <type>FastEthernet</type>
                     <identifier>FastEthernet5/1</identifier>
                     <ipInterface>
                          <ipAddress>20.10.30.45/255.255.255.0</ipAddress>
                     </ipInterface>
                  </interface>
               </interfaceList>
            <configState>committed</configState>
          </device>
</deviceConfigResponse>
```

Figure 70: GetDeviceConfigByGID Response Example

Table 61: GetDeviceConfigById Response Elements and Attributes Description

Element.Attribute Name	Definition
deviceConfigResponse	Returns the device configuration that matches the object identifier
device	The device class as described in 2.1.3.

```
<xs:element name="deviceConfigResponse" type="DeviceConfigResponse"/>
<xs:complexType name="DeviceConfigResponse">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="device" type="Device"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 71: DeviceConfigResponse XML Schema

Method specific errors (in addition, the error codes defined for the *GetGroupList* method are also applicable here) :

Code	Description
2007	This error will be returned if the requested Device GID does not exist in the CSM Server.
2008	This error will be returned if API is not able to communicate with the configuration archive module.
2009	This error will be returned if API is not able to fetch the configuration from the Configuration Archive Module
2011	This error will be returned if the user is not authorized to view the configuration.

Table 62: GetDeviceConfigByGID Method Error Codes

3.2.6 Method GetDeviceConfigByName

The GetDeviceConfigByName method returns a specific device object and its associated configuration based on the device name passed into the method. The user requesting the device configuration must have both the "view_device" and "view_cli" privilege on the device for which the config is being requested. The data in the "fullConfig" element that this method returns is equal to the data from a typical "show running-config" command exectuted on the device. This full configuration is returned from CSM's Configuration Archive (CA) component. The CA is updated during device discovery and/or deployments. **NOTE**: *This fullConfig will not contain any Out Of Band (OOB) configuration updates done directly on the device without using CSM.*

Please also see Section 3.2.11 and Section 5.

3.2.6.1 Request

An example of the method GetDeviceConfigByName request is shown in the figure below.. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getDeviceConfigByName

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<deviceConfigByNameRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <name>rtr-1.cisco.com</name>
</deviceConfigByNameRequest>
```

Figure 72: Method GetDeviceConfigByName Request Example

Table 63: Method GetDeviceConfigByName Request URL Attribute Descriptions

URL Attribute Name	Definition
deviceConfigByNameRequest	A device name request
name	The device name requested to be returned
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deviceConfigByNameRequest" type="DeviceConfigByNameRequest"/>
<xs:complexType name="DeviceConfigByNameRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="name" type="xs:string"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 73: DeviceConfigByNameRequest XML Schema

3.2.6.2 Response

The response is the same as GetDeviceConfigByGID method.

Method specific errors (in addition, the error codes defined for the *GetGroupList* and *GetDeviceConfigByGID* methods are also applicable here) :

Code	Description
2010	This error will be returned if the requested device name does not exist in the CSM Server.
2012	This error will be returned if device name is null or empty in the request.

Table 64: GetDeviceConfigByName Method Error Codes

3.2.7 Method GetPolicyListByDeviceGID

The GetPolicyListByDeviceGID method returns the list of policy names and their types, for a particular device GID. The returned list includes only the supported policy types in the current version of the API. A policy type may also not be returned if the corresponding policy is not configured. To use this API user needs to have a *view_device* RBAC privilege.

Following is the list of policies supported in this version, Version 1.0. The exact value from the policy type column need to be used in the request of the API's *GetPolicyConfigByName* and *GetPolicyConfigyByDeviceGID* methods.

Policy Type	**Description**
DeviceAccessRuleFirewallPolicy	Used to configure ACLs.
DeviceAccessRuleUnifiedFirewallPolicy	Used to configure Unified ACLs.
DeviceBGPRouterPolicy	Routing protocol.
DeviceNATTimeoutsRouterPolicy	Used for configuring Router NAT.
DeviceNATTransOptionsFirewallPolicy	Used for configuring Firewall NAT.
DeviceStaticRoutingFirewallPolicy	Used for configuring Firewall NAT.
DeviceStaticRoutingRouterPolicy	Used in configuring Router static route.
InterfaceNAT64ManualFirewallPolicy	Used for configuring NAT64 policy
InterfaceNATAddressPoolFirewallPolicy	Used for configuring Firewall NAT.
InterfaceNATDynamicRulesFirewallPolicy	Used for configuring Firewall NAT.
InterfaceNATDynamicRulesRouterPolicy	Used for configuring Router NAT.
InterfaceNATManualFirewallPolicy	Used for configuring Firewall NAT.
InterfaceNATObjectFirewallPolicy	Used for configuring Firewall NAT.
InterfaceNATPolicyDynamicRulesFirewallPolicy	Used for configuring Firewall NAT.
InterfaceNATRouterPolicy	Used for configuring Router NAT.
InterfaceNATStaticRulesFirewallPolicy	Used for configuring Firewall NAT.
InterfaceNATStaticRulesRouterPolicy	Used for configuring Router NAT.
InterfaceNATTransExemptionsFirewallPolicy	Used for configuring Firewall NAT.

3.2.7.1 Request

An example of the method GetPolicyListByDeviceGID request is shown in the figure below.. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getPolicyListByDeviceGID

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<policyListByDeviceGIDRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <gid>00000000-0000-0000-0000-000023456781</gid>
</policyListByDeviceGIDRequest>
```

Figure 74: Method GetPolicyListByDeviceGID Request Example

Table 65: Method GetPolicyListByDeviceGID Request URL Argument Descriptions

HTTP/XML Content	Definition
getPolicyListByDeviceGID	Policy list request that contains request arguments
Element: gid	An element that identifies the device the policies are configured on.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="policyListByDeviceGIDRequest" type="PolicyListByDeviceGIDRequest"/>
<xs:complexType name="PolicyListByDeviceGIDRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 75: PolicyListByDeviceGIDRequest XML Schema

3.2.7.2 Response

An example of the GetPolicyListByDeviceGID response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version="1.0" encoding="UTF-8"?>
<policyListDeviceResponse>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <policyList>
          <policyDesc>
           <name>– local --</name>
           <type>DeviceOSPFPolicy</type>
          </policyDesc>
          <policyDesc>
           <name>MySharedPolicy</name>
           <type> DeviceAccessRuleFirewallPolicy </type>
          </policyDesc>
        </policyList>
</policyListDeviceResponse>
```

Figure 76: GetPolicyListByDeviceGID Response Example

Table 66: GetPolicyListByDeviceGID Response Elements and Attributes Description

XML Element & Attributes	Definition
policyListResponse	Returns a list of 0 or more policy descriptors that are configured on the identified device
Element List: policyList	A list of policy descriptor elements
Element: policyDesc	The policy descriptor that includes its name and type. The type is usually the same name as the object name defined for that policy in the schema. For example for Firewall rules, the type will be "DeviceAccessRuleFirewallPolicy". Name of the policy, will be -- *local* -- if the policy is local/private to this device. Alternately the name will contain the user configured policy name (other than – local --) if the current policy is a *"shared"* policy (say something like "SharedAccessRule"). Shared policies are policies that are shared across multiple devices. A shared policy name is unique for a specific policy type across the whole system. For example there can only be only policy named as *SharedAccessRule* for say a *DeviceAccessRuleFirewallPolicy*. And this *SharedAccessRule* can be applied to multiple devices that support that policy.

```
        <xs:element name="policyListDeviceResponse" type="PolicyListDeviceResponse"/>
    <xs:complexType name="PolicyListDeviceResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="policyList" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="policyDesc" type="EntityDescriptor"
minOccurs="1" maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 77: PolicyListDeviceResponse XML Schema

Method specific errors :

Code	Description
1001	This error will be returned if the requested device is not be configured with any of the supported policies.
2013	This error will be returned if API is not able to fetch the internal policy list from the CSM server.

Table 67: GetPolicyListByDeviceGID Method Error Codes

3.2.8 Method GetPolicyConfigByName

The GetPolicyConfigByName method returns a specific policy object and its associated configuration based on the shared policy name passed into the method. Access to this API requires a **view_policy** privilege on the policy being queried. This method must only be used to fetch data for shared policies (i.e. policies that are not -- *local* --). The applicable shared policies for a device are available via the GetPolicyListByDeviceGID method.

3.2.8.1 Request

An example of the method GetPolicyConfigByName request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getPolicyConfigByName

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<policyConfigByNameRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <name>global FW policy-1</name>
        <policyType>DeviceAccessRuleFirewallPolicy</policyType>
</policyConfigByNameRequest>
```

Figure 78: Method GetPolicyConfigByName Request Example

Table 68: Method GetPolicyConfigByName Request URL Attribute Descriptions

Method Details	Definition
policyConfigByNameRequest	A name request and same attributes as previously defined and returned by the GetPolicyListByDeviceGID call.
Element: policyType	The policy type same as the types returned by the GetPolicyListByDeviceGID call. The list of allowable types are documented in the **GetPolicyListByDeviceGID** method section.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="policyConfigByNameRequest" type="PolicyConfigByNameRequest"/>
<xs:complexType name="PolicyConfigByNameRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
```

Figure 79: PolicyConfigByName Request XML Schema

3.2.8.2 Response

An example of the response to the GetPolicyConfigByName request is shown in the figure below. The fields in these messages are described in the table below. The response contains the policy and referenced policy object data contained in the policy. Note: The response of this method is paginated. Please see section 2.2.1.1 for further details.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ns1:policyConfigDeviceResponse xmlns:ns1="csm">
  <protVersion>1.0</protVersion>
  <policy>
    <deviceAccessRuleFirewallPolicy>
      <gid>00000000-0000-0000-0000-004294967927</gid>
      <name/>
      <lastUpdateTime>2011-08-05T05:04:49.926Z</lastUpdateTime>
      <type>FirewallRule</type>
      <orderId>1000</orderId>
      <isMandatoryAggregation>true</isMandatoryAggregation>
      <configState>deployed</configState>
      <isEnabled>true</isEnabled>
      <direction>In</direction>
      <permit>true</permit>
      <interfaceRoleObjectGIDs>
        <gid>00000000-0000-0000-0000-004294967559</gid>
      </interfaceRoleObjectGIDs>
      <sources>
        <networkObjectGIDs>
          <gid>00000000-0000-0000-0000-000000000100</gid>
        </networkObjectGIDs>
        <interfaceRoleObjectGIDs/>
      </sources>
      <destinations>
        <networkObjectGIDs>
          <gid>00000000-0000-0000-0000-000000000100</gid>
        </networkObjectGIDs>
        <interfaceRoleObjectGIDs/>
      </destinations>
      <services>
        <serviceObjectGIDs>
          <gid>00000000-0000-0000-0000-000000001041</gid>
        </serviceObjectGIDs>
      </services>
      <logOptions>
      </logOptions>
      <iosOptions>None</iosOptions>
    </deviceAccessRuleFirewallPolicy>
  </policy>
  <policyObject>
    <networkPolicyObject>
      <gid>00000000-0000-0000-0000-000000000100</gid>
      <name>any</name>
      <lastUpdateTime>2011-08-04T18:58:22.816Z</lastUpdateTime>
      <parentGID>00000000-0000-0000-0000-000000000000</parentGID>
      <type>Network</type>
      <comment>Predefined any network</comment>
      <nodeGID>00000000-0000-0000-0000-000000000001</nodeGID>
      <isProperty>false</isProperty>
      <subType/>
      <isGroup>false</isGroup>
      <ipv4Data>0.0.0.0/0.0.0.0</ipv4Data>
    </networkPolicyObject>
          ……..
  </policyObject>
</ns1:policyConfigDeviceResponse>
```

Figure 80: GetPolicyConfigByName Response Example

Element.Attribute Name	Definition
policyConfigResponse	Returns the policy configuration that matches the object identifier
Policy	The policy class as described in 3.1.1.

Table 69: GetPolicyConfigByName Response Elements and Attributes Description

```
<xs:element name="policyConfigResponse" type="PolicyConfigResponse"/>
<xs:complexType name="PolicyConfigResponse">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="policy" type="BasePolicy"/>
                <xs:element name="policyObject" type="BasePolicyObject"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
```

Figure 81: PolicyConfigResponse XML Schema

The following are error codes specific to this method. There could be additional generic errors that the method might return in cases of error

Code	Description
18	Returned if an internal error occurred while fetching config.
19	Returned if required input parameters are missing in the request.
20	Returned if the requested Policy Type is not correct.
22	Returned if the user is not authorized to view this policy data.
23	Returned if no configuration data is available for requested input parameters.
24	Returned if an internal error encountered by the server (when processing a policy type that is not supported).

Table 70: GetPolicyConfigByName Method Error Codes

3.2.9 Method GetPolicyConfigByDeviceGID

The GetPolicyConfigByDeviceGID method returns a specific policy and its associated policy objects based on the device id and policy type passed into the method. Access to this API requires a **view_policy** privilege on the policy being queried and also a **view_device** privilege on the device for which this policy is being fetched.

This method returns the applicable policy on a device (*irrespective of whether it is local or shared*). Thus this is often the most convienient method to read supported policy configuration of a device.

3.2.9.1 Request

An example of the method GetPolicyConfigByDeviceGID request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/configservice/getPolicyConfigById

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<policyConfigByDeviceGIDRequest>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <gid>00000000-0000-0000-0000-004294967927</gid>
        <policyType>DeviceAccessRuleFirewallPolicy</policyType>
</policyConfigByDeviceGIDRequest>
```

Figure 82: Method GetPolicyConfigByDeviceGID Request Example

Table 71: Method GetPolicyConfigByDeviceGID Request URL Attribute Descriptions

Method Details	Definition
policyConfigByDeviceByGIDRequest	A global ID request and same attributes as previously defined
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="policyConfigByDeviceGIDRequest" type="PolicyConfigByDeviceGIDRequest"/>
<xs:complexType name="PolicyConfigByDeviceGIDRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="gid" type="ObjectIdentifier"/>
                <xs:element name="policyType" type="string"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 83: PolicyConfigByDeviceGIDRequest XML Schema

3.2.9.2 Response

The response of this method including the schema and error codes are the same as the *GetPolicyConfigByName* method described previously.

3.2.10 Method GetSharedPolicyNamesByType

This method returns the list of all the Shared Policies present in the Policy View of CSM for a given policy type.

3.2.10.1 REST Request:

```
URL:
http://hostname/nbi/configservice/getSharedPolicyListByType

HTTP Header:
Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT;
path=/; domain=.hostdomain.com

XML Argument:
<?xml version=”1.0” encoding=”UTF-8”?>
< policyNamesByTypeRequest>
                <protVersion>1.0</protVersion>
                <reqId>123</reqId>
                <policyType>DeviceAccessRuleFirewallPolicy</policyType>
</ policyNamesByTypeRequest >
```

Figure 84: getSharedPolicyNamesByType Request Example

Table 72: Method getSharedPolicyNamesByType Request URL Attribute Descriptions

Method Details	Definition
policyNamesByTypeRequest	A request to get the list of all the Shared Policies defined in CSM for a given policy Type.

Method Details	Definition
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

3.2.10.2 Response Object:

```
<?xml version="1.0" encoding="UTF-8"?>
<policyNamesResponse>
        <protVersion>1.0</protVersion>
        <reqId>123</reqId>
        <policyType>DeviceAccessRuleFirewallPolicy</policyType>
        <policy>
          <name>policy-1</name>
          <deviceAssignments>
              <device>
                 <deviceGID>00000000-0000-0000-0000-004294967308</deviceGID>
                 <deviceName>1.1.1.1</deviceName>
              </device>
        </deviceAssignments>
        </policy>
        <policy>
          <name>policy-2</name>
          <deviceAssignments>
              <device>
                 <deviceGID>00000000-0000-0000-0000-004294967309</deviceGID>
                 <deviceName>1.1.1.2</deviceName>
              </device>
          </deviceAssignments>
        </policy>
```

Figure 85: GetSharedPolicyNamesByType Response Example

Element.Attribute Name	Definition
policyNamesResponse	Returns the policy Names and the Device Assignments for the Policy Type passed in the Request.
policyNamesResponse.policyType	The Policy Type passed in the Request.

Element.Attribute Name	Definition
policyNamesResponse.policy	A policy defined in CSM for the Requested Policy Type.
policyNamesResponse.policy.name	Name of the Shared Policy.
policyNamesResponse.policy.deviceAssignments	List of devices the Shared Policy is assigned to.
policyNamesResponse.policy.deviceAssignments.deviceGID	Device GID of the Device the Shared Policy is assigned to.
policyNamesResponse.policy.deviceAssignments.deviceName	Device Name of the Device the Shared Policy is assigned to.

Table 73: GetSharedPolicyNamesByType Response Elements and Attributes Description

```
<xs:element name="policyNamesResponse" type="PolicyNamesResponse"/>
<xs:complexType name="PolicyNamesResponse">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="policyTYpe" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="policy" minOccurs="1" maxOccurs="unbounded">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="policyName" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                                          <xs:element name="deviceAssignments" minOccurs="0"
maxOccurs="1">
                                                <xs:complexType>
                                                      <xs:sequence>
                                                            <xs:element name="device" type="xs:string"
minOccurs="1" maxOccurs="unbounded”>
                                                                  <xs:complexType>
                                                                        <xs:sequence>
                                                                              <xs:element name="deviceGID"
type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                                                                              <xs:element name="deviceName"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                                                                        </xs:sequence>
                                                                  </xs:complexType>
                                                            </xs:element>
                                                      </xs:sequence>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                              </xs:complexType>
```

Figure 86: PolicyNamesResponse XML Schema

3.2.11 Method CreateCSMSession

This method will create a CSMSession to be used with other write api methods. Internally it will create a ticket or activity based on the mode of the CSM Server. All csm server modes supports this method. For detailed information about it’s usage please see section Using CSMSession and Write APIs

3.2.11.1 Request

```
URL:

https://hostname/nbi/configservice/createCSMSession

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 2-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:


<?xml version="1.0" encoding="UTF-8"?>
<p:newCSMSessionRequest xmlns:p="csm">
<csmSessionDescription>creating CSM API session to test</csmSessionDescription>
</p:newCSMSessionRequest>
```

Figure 87: Method CreateCSMSession Request Example

Table 74:Method CreateCSMSession Request URL Attribute Descriptions

HTTP/XML Content	Definition
newCSMSessionRequest	CSM Session request that contains request arguments
Element: csmSessionName	An element that provides a string value to use while creating the CSM Session. This could be a ticketId or activity name.
Element: csmSessionDescription	An element that provides description (optionally) specifying the requirement for creating a new CSMSession
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="newCSMSessionRequest" type="CSMSessionRequest" />
<xs:complexType name="CSMSessionRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionName" type="xs:string" minOccurs="0"
                              maxOccurs="1" />
                        <xs:element name="csmSessionDescription" type="xs:string"
                              minOccurs="0" maxOccurs="1" />
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Figure 88: CreateCSMSessionRequest XML Schema

3.2.11.2 Response

The response of this method returns the unique csmSessionId that will be used in all write api related methods. The following figure gives the example and the xsd of the response.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:csmSessionResponse xmlns:ns1="csm">
     <protVersion>1.0</protVersion>
     <csmSessionGID>00000000-0000-0000-0000-017179869189</csmSessionGID>
</ns1:csmSessionResponse>
```

Figure 89: CreateCSMSession Response Example

Table 75: CreateCSMSession Response Elements and Attributes Description

XML Element & Attributes	Definition
csmSessionResponse	Returns CSMSession details
Element List: csmSessionGID	Unique value that should use for further write operations

```
      <xs:element name="csmSessionResponse" type="CSMSessionResponse" />
      <xs:complexType name="CSMSessionResponse">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="csmSessionGID" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 90: CreateCSMSession Response XSD

3.2.12 Method ValidateCSMSession

This method is used to validate the changes done in a CSMSession. Based on the validation results submit of that CSMSession can be done. For more details on the CSMSession please see section Using CSMSession and Write APIs.

3.2.12.1 Request

The validation is done for the CSMSession that is uniquely identified by the csmSessionGID which is given in the request parameter. The request example and the XSD is given in the below figures.

```
Method : POST

URL : https://hostname/nbi/configservice/validateCSMSession

Headers : Content-Type:text/xml

Body :

<?xml version="1.0" encoding="UTF-8"?>
<ns1:csmSessionOperationRequest xmlns:ns1="csm">
 <csmSessionGID>00000000-0000-0000-0000-068719477096</csmSessionGID>
</ns1:csmSessionOperationRequest>
```

Figure 91: validateCSMSession Request Example

Table 76: Method ValidateCSMSession Request URL Attribute Descriptions

HTTP/XML Content	Definition
csmSessionOperationRequest	CSM Session request that contains request arguments
Element: csmSessionName	An element that provides the name of the csmSession using which the session was created.
Element: csmSessionGID	An element that provides the unique GID of the csmSession.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="csmSessionOperationRequest" type="CSMSessionOperationRequest"/>
<xs:complexType name="CSMSessionOperationRequest">
      <xs:complexContent>
            <xs:extension base="BaseReqResp">
                  <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:choice id="select">
                              <xs:element name="csmSessionGID" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />
                              <xs:element name=" csmSessionName" type="xs:string" minOccurs="1"
                                    maxOccurs="1" />
                        </xs:choice>
                        <xs:element name="comments" type="xs:string" minOccurs="0"
                              maxOccurs="1" />
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Figure 92: validateCSMSession Request XML Schema

3.2.12.2 Response

The response contains the validation messages for the changes that have been made in that particular CSMSession. The following gives the example and the XSD of the response.

```
<ns1:csmSessionValidationResponse>
<protVersion>1.0</protVersion>
<statusCode>SUCCESS</statusCode>
<validationMessage>CSM Session Validation successful</validationMessage>
<validationResults>
   <validationResult>
      <deviceGID>
         <gid>00000000-0000-0000-0000-004294967310</gid>
      </deviceGID>
    <validationDetails>
      <title>
         FWSVC Access Rules Warning in Rule ASA warning - more details
      </title>
      <severity>Warning</severity>
      <description>
          FWSVC Access Rules Warnings -> The following interface roles inside ,test-ether , do
          not exist on device. Rules for these interfaces will not be generated on device
      </description>
      <action>
         Please specify Interface Roles in Rule which match those on device
      </action>
    </validationDetails>
   </validationResult>
</validationResults>
</ns1:csmSessionValidationResponse>
```

Figure 93: validateCSMSession Response Example

Table 77: ValidateCSMSession Response Elements and Attributes Description

XML Element & Attributes	Definition
csmSessionValidateResponse	An element that mentions the details of the validation results.
statusCode	An element that describes the status of the whole operation.
approvalRequired	An element that mentions if approval is required for this session. This is based on the mode of the CSM Server and the admin options configured. If this is set to “True”, after submitCSMSession, ApproveCSMSession has to be invoked seperately.
Element : validationResults	An element have the validation result status.
Element List: deviceGID	Unique value that specifies the device id.
Element: ValidationDetails	This element has all the information on the individual validation messages.

```
      <xs:element name="csmSessionValidationResponse" type="CSMSessionValidationResponse" />
    <xs:complexType name="CSMSessionValidationResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="statusCode" type="OperationStatus" minOccurs="1" maxOccurs="1" />
           <xs:element name=""approvalRequired" type="xs:boolean" minOccurs="1" maxOccurs="1" />
            <xs:element name="validationMessage" type="xs:string" minOccurs="1" maxOccurs="1" />
            <xs:element name="validationResults" type="ValidationResults" minOccurs="0"
maxOccurs="unbounded" />
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
    </xs:complexType>

   <xs:complexType name="ValidationResults">
        <xs:sequence minOccurs="1" maxOccurs="unbounded">
          <xs:element name="validationResult" type="ValidationResult" minOccurs="0" maxOccurs="unbounded"
/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ValidationResult">
        <xs:sequence minOccurs="1" maxOccurs="unbounded">
           <xs:element name="deviceGID" type="ObjectIdentifierList" minOccurs="0" maxOccurs="unbounded" />
           <xs:element name="validationDetails" type="ValidationDetails" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ValidationDetails">
        <xs:sequence minOccurs="1" maxOccurs="unbounded">
            <xs:element name="title" type="xs:string" minOccurs="1"
                maxOccurs="1" />
            <xs:element name="severity" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="Critical" />
                        <xs:enumeration value="Warning" />
                        <xs:enumeration value="Info" />
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="description" type="xs:string" minOccurs="1"
                maxOccurs="1" />
            <xs:element name="action" type="xs:string" minOccurs="0"
                maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>
```

Figure 94: validateCSMSession Response XML Schema

3.2.13 Method SubmitCSMSession

To submit the changes made during a CSMSession use this method. The submitCSMSession internally does a validate and only on a successful validation submit is carried out.

3.2.13.1 Request

The input to the submitCSMSession takes in the unique CSMSessionGID and does the validate and submit of the changes carried out in that particular CSMSession. The following gives the details about the request example and the XSD.

```
Method : POST

URL : https://hostname/nbi/configservice/submitCSMSession

Headers : Content-Type:text/xml

Body :

<?xml version="1.0" encoding="UTF-8"?>
<csm:submitCSMSessionRequest xmlns:csm="csm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<csmSessionGID>00000000-0000-0000-0000-068719477096</csmSessionGID>
<submitComments>Submission test</submitComments>
<continueOnWarnings>true</continueOnWarnings>
</csm:submitCSMSessionRequest>
```

Figure 95: SubmitCSMSession Request Example

Table 78: SubmitCSMSession Request URL Attributes Description

XML Element & Attributes	Definition
submitCSMSessionRequest	An element that mentions the details of the submit operation.
Element : csmSessionGID	Unique value that specifies the CSMSession ID for which the changes to be validated
Element: submitComments	Any comments to be saved
Element: continueOnwarnings	Boolean value true – to continue submit on warning message False – abort submit on detecting warnings

3.2.13.2 Response

The response is the same as that of validateCSMSession.

3.2.14 Method DiscardCSMSession

This method can be used to discard the changes that were made in a CSMSession.

3.2.14.1 Request

The request is same as in validateCSMSession. The following only gives an example of the method, the rest of details are similar to validateCSMSession.

```
Method : POST

URL : https://hostname/nbi/configservice/discardCSMSession

Headers : Content-Type:text/xml

Body :

<?xml version="1.0" encoding="UTF-8"?>
<ns1:csmSessionOperationRequest xmlns:ns1="csm">
  <csmSessionGID>00000000-0000-0000-0000-068719477096</csmSessionGID>
</ns1:csmSessionOperationRequest>
```

Figure 96: DiscardCSMSession XML Request Example.

3.2.14.2 Response

The response gives the status of the discarded CSMSession and its csmSessionGID. The following gives the example and the XSD of the response.

```
<ns1:csmSessionResultResponse>
<protVersion>1.0</protVersion>
<result>CSM Session discarded successfully!</result>
<csmSessionGID>00000000-0000-0000-0000-008589935876</csmSessionGID>
</ns1:csmSessionResultResponse>
```

Figure 97: DiscardCSMSession Response XML Example

Table 79: DiscardCSMSession Response URL Attributes and Elements Description

XML Element & Attributes	Definition
csmSessionResultResponse	An element that mentions the results of the CSMSession operation
Element List: csmSessionGID	Unique value that specifies the CSMSession ID for which the changes are discarded
ElementList:result	An element that tells the success or failure

```
<xs:element name="CSMSessionResultResponse" type="CSMSessionResultResponse" />
    <xs:complexType name="CSMSessionResultResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="result" type="xs:string" minOccurs="1"
                        maxOccurs="1" />
                    <xs:element name="CSMSessionGID" type="ObjectIdentifier"
                        minOccurs="1" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 98: DiscardCSMSession response XSD.

3.2.15 Method ApproveCSMSession

This method is used to approve or reject the CSMSession that has been submitted. This method needs to be used only in the case where the CSM Server is in workflow-mode. In all other modes submitCSMSession does auto-approval of the CSMSession. The submitCSMSession method will notify whether approval is needed or not.

3.2.15.1 Request

The following gives an example of the request and it's XSD.

```
Method : POST

URL : https://hostname/nbi/configservice/approveCSMSession

Headers : Content-Type:text/xml

Body :

<?xml version="1.0" encoding="UTF-8"?>
<ns1:csmApproveOperationRequest xmlns:ns1="csm">
<csmSessionGID>00000000-0000-0000-0000-184683593741</csmSessionGID>
<approvalStatus>true</approvalStatus>
</ns1:csmApproveOperationRequest>

Note : Approval Status false for Rejecting the activity. It can be used in workflow enabled
mode.
```

Figure 99: ApproveCSMSession Request XML Example

Table 80: ApproveCSMSession request URL elements and attributes Description

XML Element & Attributes	Definition
csmApproveOperationRequest	An element that mentions the details of the CSMApproveOperation request
Element: csmSessionGID	Unique value that user can provide the CSMSessionGID for which the changes to be validated
Element:csmSessionName	Unique value that user can provide the name for which the changes to be validated
Element:comments	Optionally any comments can be specified
Element: approvalStatus	Approval status , Either true or false to approve or reject the change list

```
<xs:element name="csmApproveOperationRequest" type="CSMApproveOperationRequest" />
      <xs:complexType name="CSMApproveOperationRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:choice id="approveselect">
                                    <xs:element name="csmSessionGId" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                                    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
                              </xs:choice>
                              <xs:element name="comments" type="xs:string" minOccurs="0" maxOccurs="1" />
                              <xs:element name="approvalStatus" type="xs:boolean"       minOccurs="1"
maxOccurs="1" />
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 100: ApproveCSMSession request XSD

3.2.15.2 Response

The response of the approveCSMSession returns the approval status and the CSMSessionGID of the session.

```
<ns1:csmApproveResultResponse>
<protVersion>1.0</protVersion>
<result>CSMSession ID Approve successfully!</result>
<csmSessionGID>00000000-0000-0000-0000-008589935912</csmSessionGID>
</ns1:csmApproveResultResponse>
```

Figure 101: ApproveCSMSession Response XML Example

Table 81: ApproveCSMSession Response Elements and Attribute Description

XML Element & Attributes	Definition
csmApproveResultResponse	An element that mentions the results of the CSMApproveResult Response
Element List: csmSessionGID	Unique value that specifies the CSMSession ID for which the changes are discarded
ElementList:result	An element that tells the success or failure

```
<xs:element name="csmApproveResultResponse" type="CSMApproveResultResponse" />
    <xs:complexType name="CSMApproveResultResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="result" type="xs:string" minOccurs="1"
                        maxOccurs="1" />
                    <xs:element name="csmSessionGId" type="ObjectIdentifier"
                        minOccurs="1" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 102: ApproveCSMSession Response XSD.

3.2.16 Method OpenCSMSession

This method can be used to open a CSMSession that was either closed earlier or the https session got timed out.

3.2.16.1 Request

The request of this method is same as that of validateCSMSession. The following gives an example of openCSMSession only.

```
Method : POST

URL : https://hostname/nbi/configservice/openCSMSession

Headers : Content-Type:text/xml

Body :

<?xml version="1.0" encoding="UTF-8"?>
<ns1:csmSessionOperationRequest xmlns:ns1="csm">
 <csmSessionGID>00000000-0000-0000-0000-068719477096</csmSessionGID>
</ns1:csmSessionOperationRequest>
```

Figure 103: OpenCSMSession request XML example.

3.2.16.2 Response

The response of this method is same as that of DiscardCSMSession.

3.2.17 Method CloseCSMSession

This method is used to close a CSMSession. The closed CSMSession can be opened using the openCSMSession.

3.2.17.1 Request

The request of this method is same as that of validateCSMSession. The following gives an example of closeCSMSession.

```
Method : POST

URL : https://hostname/nbi/configservice/closeCSMSession

Headers : Content-Type:text/xml

Body :

<?xml version="1.0" encoding="UTF-8"?>
<ns1:csmSessionOperationRequest xmlns:ns1="csm">
 <csmSessionGID>00000000-0000-0000-0000-068719477096</csmSessionGID>
</ns1:csmSessionOperationRequest>
```

Figure 104: CloseCSMSession request XML example.

3.2.17.2 Response

The response of this method is same as that of discardCSMSession.

3.2.18 Method AddPolicyObject

This method can be used to add a new policy object. The supported list of policy objects are mentioned in section 1.3.1. Modify priviledge is needed on the particular object to add the object. It also supports addition of heterogeneous bulk objects in a single call. A single call of the API happens within a single transaction and hence if addition of one of policy objects fails all previous successful insertions would be reverted back. This will behave in fail-fast mode. To understand the individual element details please go through section 3.1.2 BasePolicyObject

This method should be used in the following use-cases:

1. Create a new building block :-
   The data part of the supported building block is passed in the addPolicyObject request.
2. Create a overridden building block:
   In CSM the overridden building block is stored as a separate building block with separate id. When creating a overridden building block for a device, then addPolicyObject should be used with node id as the device ID and parent ID as the id of building block for which overridden value is created. Also all the data part should be sent with the request, where some values would differ from the parent building block. The parent building block's "isProperty" should be set to true to allow overrides.

3.2.18.1 Request

The request takes in the policy objects and the CSMSession used to make this change. The following gives an example and the XSD of the request.

```
URL:

https://hostname/nbi/configservice/addPolicyObject

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<csm:addPolicyObjectRequest xmlns:csm="csm">
   <csmSessionGID>00000000-0000-0000-0000-425201762587</csmSessionGID>
   <enforceDuplicateDetection>false</enforceDuplicateDetection>
   <networkPolicyObject>
      <name>27_NO_1</name>
      <parentGID>00000000-0000-0000-0000-000000000000</parentGID>
      <updatedByUser>dkakoll</updatedByUser>
      <lastCommitTime>2014-02-27T04:53:09.36Z</lastCommitTime>
      <ticketId>dkakoll_27.Feb.2014_10.22.05</ticketId>
      <type>NetworkPolicyObject</type>
      <comment> </comment>
      <nodeGID>00000000-0000-0000-0000-000000000001</nodeGID>
      <isProperty>false</isProperty>
      <subType></subType>
      <isGroup>false</isGroup>
      <ipData>1.19.0.3</ipData>
      <ipData>1.19.0.4</ipData>
   </networkPolicyObject>
</csm:addPolicyObjectRequest>
```

Figure 105: addPolicyObject URL Request Example

Table 82: addPolicyObject Request Elements and Attributes Description

HTTP/XML Content	Definition
addPolicyObjectRequest	Add policy object containing request arguments
Element: csmSessionGID	An element that uniquely defines a CSMSession
Element: enforceDuplicateDetection	if Redundant/Duplicate object is detected, throw error (set: true) or continue (set: false) with saving.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="addPolicyObjectRequest" type="ManagePolicyObjectRequest" />
<xs:complexType name="ManagePolicyObjectRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="CSMSessionGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1" />
                    <!-- if Redundant/Duplicate object is detected, throw warning (set: true) or continue
(set: false) with saving -->
                    <xs:element name="enforceDuplicateDetection" type="xs:boolean" minOccurs="0"
maxOccurs="1" />
                    <!-- ALL Supported POLICY OBJECT TYPES -->
                    <xs:element name="networkPolicyObject" type="NetworkPolicyObject" minOccurs="0"
maxOccurs="500" />
                    <xs:element name="servicePolicyObject" type="ServicePolicyObject" minOccurs="0"
maxOccurs="500" />
                    <xs:element name="interfaceRolePolicyObject" type="InterfaceRolePolicyObject"
minOccurs="0" maxOccurs="500" />
                    <xs:element name="timeRangePolicyObject" type="TimeRangePolicyObject"
minOccurs="0" maxOccurs="500" />
                    <xs:element name="identityUserGroupPolicyObject"
type="IdentityUserGroupPolicyObject" minOccurs="0" maxOccurs="500" />
                    <xs:element name="securityGroupPolicyObject" type="SecurityGroupPolicyObject"
minOccurs="0" maxOccurs="500" />
                    <xs:element name="portListPolicyObject" type="PortListPolicyObject" minOccurs="0"
maxOccurs="500" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

Figure 106: AddPolicyObject request XSD.

3.2.18.2 Response

The response will give the details of the created object id and it's type. If any failure then the details of the failed object would be sent. Following gives an example of the response and its xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:addPolicyObjectResponse xmlns:ns1="csm">
  <protVersion>1.0</protVersion>
   <message>CREATE Policy Object Operation Successful!</message>
   <resultObject>
     <objectGID>00000000-0000-0000-0000-425201762588</objectGID>
     <name>27_NO_1</name>
     <type>Network</type>
   </resultObject>
</ns1:addPolicyObjectResponse>
```

Figure 107: AddPolicyObject XML response example

Table 83: AddPolicyObject Response XML Attribute and Element Description

HTTP/XML Content	Definition
addPolicyObjectResponse	Add policy object containing request arguments
Element: message	The overall status of the operation.
Element: resultObject	The individual results of object.
Element: objectGID	An element that uniquely identifies the ID of the policy object that we added.
Element: name	An element that gives the name of the policy object.
Element: type	An element that provides the type of the policy object.
Element: info	Any other details about the object if needed.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="addPolicyObjectResponse" type="ManagePolicyObjectResponse" />
      <xs:complexType name="ManagePolicyObjectResponse">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="message" type="xs:string" minOccurs="0" maxOccurs="1" />
                              <xs:element name="resultObject" maxOccurs="unbounded" minOccurs="0">
                                    <xs:complexType>
                                          <xs:sequence>
                                                <xs:element type="xs:string" name="objectGID" minOccurs="1"
maxOccurs="1" />
                                                <xs:element type="xs:string" name="name" minOccurs="1"
maxOccurs="1" />
                                                <xs:element type="xs:string" name="type" minOccurs="1"
maxOccurs="1" />
                                                <xs:element type="xs:string" name="info" minOccurs="0"
maxOccurs="1" />
                                          </xs:sequence>
                                    </xs:complexType>
                              </xs:element>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 108: AddPolicyObject Response XSD

3.2.19 Method ModifyPolicyObject

Modify policy object can be used to modify the data part of the policy object. This can also be used to rename the policy objects. Policy Object is identified uniquely by the gid that is sent with the request. Even if one field needs to be updated, all the data needs to be sent with the request. Ideally this is used after the policy object is read and then the required field updated and sent as input to the modify policy object method. This also takes in hetrogenous and homogeneous bulk request and works in a single transaction and in fail-fast mode.

In the following cases this method can be used:

1. Modifying a non-overridden Building Block:
   The “gid” element that is passed in the request defines the building block that will be modified. So this method can be used also to rename the building block. Along with the request the complete data is required to be passed with whatever elements that needs to be modified.

2. Modifying a overridden Building Block:
   To modify some data of a overridden BB for a particular device, then the gid of that particular building block needs to sent in the request and not the parent building blocks gid. If only the name is known of the building block, then use the method getPolicyObject to get the overridden building blocks gids and use it in this method.

3.2.19.1 Request

The request of modify policy object takes in the type “ManagePolicyRequest” which is the same for addPolicyObject. Following gives an example and the XSD.

```
URL:

https://hostname/nbi/configservice/modifyPolicyObject

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version=”1.0” encoding=”UTF-8”?>
<csm:modifyPolicyObjectRequest xmlns:csm="csm">
        <csmSessionGID>00000000-0000-0000-0000-240518168641</csmSessionGID>
        <enforceDuplicateDetection>false</enforceDuplicateDetection>

        <servicePolicyObject>
                 <gid>00000000-0000-0000-0000-231928234391</gid>
                 <name>01_NbAPI_SPO</name>
                 <type>ServicePolicyObject</type>
                 <comment>Modified via NbAPI </comment>
                 <isProperty>false</isProperty>
                 <subType>SO</subType>
                 <isGroup>false</isGroup>
                 <serviceParameters>
                         <protocol>ip</protocol>
                         <icmpMessage/>
                 </serviceParameters>
        </servicePolicyObject>

</csm:modifyPolicyObjectRequest>
```

Figure 109: ModifyPolicyObject Request Example

```
<xs:element name="modifyPolicyObjectRequest" type="ManagePolicyObjectRequest" />
```

Figure 110: ModifyPolicyObject Request XSD

3.2.19.2 Response

The response is also the same as that of addPolicyObject. Following gives the example and the XSD of the response.

```
<ns1:modifyPolicyObjectResponse xmlns:ns1="csm">
        <protVersion>1.0</protVersion>
        <message>MODIFY Policy Object Operation Successful!</message>
        <resultObject>
                 <objectGID>00000000-0000-0000-0000-231928234391</objectGID>
                <name>01_NbAPI_SPO</name>
                 <type>Service</type>
        </resultObject>
</ns1:modifyPolicyObjectResponse>
```

Figure 111: ModifyPolicyObject Response Example

```
<xs:element name="modifyPolicyObjectResponse" type="ManagePolicyObjectResponse" />
```

Figure 112: ModifyPolicyObject Response XSD.

3.2.20 Method DeletePolicyObject

This method is used to delete the policyobject given it's unique GID. This supports bulk delete and happens in a single transaction, and works in fail fast mode. The method can be used in the following scenarios:

1. Delete a Building Block:
   The gid of the policy object along with the type of the object needs to be passed.
2. Delete a override:
   If for a particular device the overridden value is to be removed and the global value needs to be retained the gid of the overridden building block needs to be passed on to this defect.

3.2.20.1 Request

The request of deletePolicyObject is also of type "`ManagePolicyObjectRequest`". But it is sufficient to mention the type and id of the policy object. The following gives an example request and the xsd.

```
URL:

https://hostname/nbi/configservice/deletePolicyObject

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:


<csm:deletePolicyObjectRequest  xmlns:csm="csm">
        <csmSessionGID>00000000-0000-0000-0000-231928234184</csmSessionGID>

        <servicePolicyObject>
          <gid>193273528324</gid>
        </servicePolicyObject>
        <timeRangePolicyObject>
          <gid>231928234007</gid>
        </timeRangePolicyObject>
        <timeRangePolicyObject>
          <gid>231928234091</gid>
        </timeRangePolicyObject>

</csm:deletePolicyObjectRequest>
```

Figure 113: DeletePolicyObject Request XML Example

```
<xs:element name="deletePolicyObjectRequest" type="ManagePolicyObjectRequest" />
```

Figure 114: DeletePolicyObject Request XSD

3.2.20.2 Response

The response is of type “ManagePolicyObjectResponse”, which is the same for add and modify policy object. If a bulk deletion fails then the error message will have the details of the failed building block. The following gives an example of response and the XSD.

```
<ns1: deletePolicyObjectResponsexmlns:ns1="csm">
    <protVersion>1.0</protVersion>
    <message>DELETE Policy Object Operation Successful!</message>
</ns1: deletePolicyObjectResponse>
```

Figure 115: DeletePolicyObject Response Example

```
<xs:element name="deletePolicyObjectResponse" type="ManagePolicyObjectResponse" />
```

Figure 116: DeletePolicyObject Response XSD

3.2.21 Method GetPolicyObject

This method can be used to retrieve the building block information by it’s name and type. It supports bulk inputs of hetrogenous types. This also gives the details of the overridden BB details along with the node id which is the device id on which it has overridden values. To read the overridden value on a particular device then the gid of that particular building block needs to be used in this method.

3.2.21.1 Request

The request takes in the type and the name or id of the building block. The request is of type “ManagePolicyObjectRequest”. Following gives an example of the request and the XSD.

```
URL:

https://hostname/nbi/configservice/getPolicyObject


<?xml version="1.0" encoding="UTF-8"?>
<p:getPolicyObjectRequest xmlns:p="csm">
<networkPolicyObject>
   <name>0_test</name>
</networkPolicyObject>
</p:getPolicyObjectRequest>
```

Figure 117: GetPolicyObject Request Example

```
<xs:element name="getPolicyObjectRequest" type="ManagePolicyObjectRequest"/>
```

Figure 118: GetPolicyObject Request XSD

3.2.21.2 Response

The response contains the details of the policy object. It includes just the details of the particular BB and gives the references of the overrides.

```
<?xml version="1.0" encoding="UTF-8"?>
   <ns1:policyObjectConfigResponse xmlns:ns1="csm">
    <protVersion>1.0</protVersion>
    <policyObject>
     <networkPolicyObject>
      <gid>00000000-0000-0000-0000-322122547219</gid>
      <name>0_test</name>
      <lastUpdateTime>1970-01-01T00:00:00Z</lastUpdateTime>
      <parentGID>00000000-0000-0000-0000-000000000000</parentGID>
      <type>NetworkPolicyObject</type>
      <comment> </comment>
      <nodeGID>00000000-0000-0000-0000-000000000001</nodeGID>
      <isProperty>true</isProperty>
      <subType>NN</subType>
      <isGroup>false</isGroup>
     </networkPolicyObject>
    </policyObject>
    <overrideDetails>
       <objectGID>00000000-0000-0000-0000-326417514595</objectGID>
       <parentGID>00000000-0000-0000-0000-322122547219</parentGID>
       <deviceGID>00000000-0000-0000-0000-034359738454</deviceGID>
    </overrideDetails>
    </ns1:policyObjectConfigResponse>
```

Figure 119: GetPolicyObject Response XML Example

Table 84: GetPolicyObject Response XML Attribute and Element Description

HTTP/XML Content	Definition
policyObjectConfigResponse	Get policy object containing response arguments
Element: csmSessionGID	An element that uniquely defines a CSMSession
Element: enforceDuplicateDetection	if Redundant/Duplicate object is detected, throw error (set: true) or continue (set: false) with saving.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session

HTTP/XML Content	Definition
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="policyObjectConfigResponse" type="PolicyObjectConfigResponse" />
      <xs:complexType name="PolicyObjectConfigResponse">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="objectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"
/>
                              <xs:element name="results" type="xs:string" minOccurs="0"
                                    maxOccurs="1" />
                              <xs:element name="details" type="xs:string" minOccurs="0"
                                    maxOccurs="1" />
                              <xs:element name="policyObject" minOccurs="1" maxOccurs="1">
                                    <xs:complexType>
                                          <xs:sequence minOccurs="1" maxOccurs="1">
                                                <xs:element name="networkPolicyObject"
type="NetworkPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="portListPolicyObject" type="PortListPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="servicePolicyObject" type="ServicePolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="interfaceRolePolicyObject"
type="InterfaceRolePolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="timeRangePolicyObject"
type="TimeRangePolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="slaMonitorPolicyObject"
type="SLAMonitorPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="aclPolicyObject" type="ACLPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="stdAcePolicyObject"
type="StandardACEPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="extendedACEPolicyObject"
type="ExtendedACEPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                                <xs:element name="identityUserGroupPolicyObject"
                                                      type="IdentityUserGroupPolicyObject" minOccurs="0"
maxOccurs="unbounded" />
                                                <xs:element name="securityGroupPolicyObject"
type="SecurityGroupPolicyObject"
                                                      minOccurs="0" maxOccurs="unbounded" />
                                          </xs:sequence>
                                    </xs:complexType>
                              </xs:element>
                        <xs:element name="overrideDetails" minOccurs="0" maxOccurs="unbounded">
                                    <xs:complexType>
                                          <xs:sequence>
                                             <xs:element name="objectGID" type="ObjectIdentifier"minOccurs="1"
maxOccurs="1" />
                                                <xs:element name="parentGID" type="ObjectIdentifier"minOccurs="1"
maxOccurs="1" />
                                                   <xs:element name="deviceGID" type="ObjectIdentifier"minOccurs="1"
maxOccurs="1" />
                                          </xs:sequence>
                                    </xs:complexType>
                              </xs:element>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 120: GetPolicyObject Response XSD

3.2.22 Method GetPolicyObjectByGID

This method can be used to retrieve the object details only from the gid of the object. This method supports bulk inputs.

3.2.22.1 Request

The request just takes the gid of the object and retrives the commited data of that BB. But if csmSessionId is given along with the request then the particular sessions information is retrieved. Following gives an example of the request and the XSD.

```
URL:

https://hostname/nbi/configservice/getPolicyObjectByGID

<?xml version="1.0" encoding="UTF-8"?>
<p:getPolicyObjectByGID xmlns:p="csm">
    <gid>322122547219</gid>
    <gid>326417514521</gid>
</p:getPolicyObjectByGID>
```

Figure 121: GetPolicyObjectByGID request example.

Table 85: GetPolicyObjectByGID Response XML Attribute and Element Description

HTTP/XML Content	Definition
getPolicyObjectByGID	Gets the policy object containing response arguments
Element: csmSessionGID	An element that uniquely defines a CSMSession
Element**:** gid	Determines uniquely a given building block.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:complexType name="GetPolicyObjectByGID">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
  <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1" />
        <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 122: GetPolicyObjectByGID request XSD.

3.2.22.2 Response

The response is same as that of getPolicyObject.

3.2.23 Method GetListofDeployableDevices

This method can be used to get the list of devices that are deployable for the current user. The device information that is got in this API is used to trigger the deployment using deployConfigByGID.

3.2.23.1 Request

The request for the method doesn't have any new element but just the session cookie. The following gives an example of the request and the XSD.

```
URL:

https://hostname/nbi/configservice/getListOfDeployableDevices

<?xml version="1.0" encoding="UTF-8"?>
  <ns1:deployableDevicesListRequest xmlns:ns1="csm">
    <protVersion>1.0</protVersion>
    <reqId>123</reqId>
  </ns1:deployableDevicesListRequest>
```

Figure 123: GetListofDeployableDevices XML Request Example

Table 86: GetListofDeployableDevices XML Element and Atrribute Description

HTTP/XML Content	Definition
deployableDevicesListRequest	This contains the request arguments
Element: sessionId	An element that required for this request is logged in session information, this should be passed as part of cookie

HTTP/XML Content	Definition
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:complexType name="DeployableDevicesListRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 124: GetListofDeployableDevices Request XSD

3.2.23.2 Response

The response has the details of the devices that have changes that are yet to be deployed i.e dirty devices. The following gives an example response and its XSD.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <ns1:deviceListResponse xmlns:ns1="csm">
  <protVersion>1.0</protVersion>
  <reqId>123</reqId>
 <deviceId>
  <deviceCapability>firewall</deviceCapability>
  <deviceName>4.4.4.4</deviceName>
  <ipv4Address>4.4.4.4</ipv4Address>
  <sysObjectID>1.3.6.1.4.1.9.1.670</sysObjectID>
  <gid>00000000-0000-0000-0000-004294968099</gid>
 </deviceId>
 </ns1:deviceListResponse>
```

Figure 125: GetListofDeployableDevices Response Example

Table 87: GetListofDeployableDevices Response XML Element and Attribute Description

XML Element & Attributes	Definition
deviceListResponse	Returns a list of 0 or more devices that match the filter parameter passed in the method
Element List: DeviceId	A list of device ID elements
Attribute: gid	The gid attribute of the device
Element: deviceCapability	One or more capabilities of the device (mandatory)
Element: ipv4Address	The IPv4 address of the device (optional)
Element: deviceName	Display name of the device
Element: sysobjectId	System object id of the device
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deviceListResponse" type="DeviceListResponse" />
    <xs:complexType name="DeviceListResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="deviceId" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence minOccurs="1" maxOccurs="1">
                                <xs:element name="deviceCapability" type="DeviceCapability"
                                    minOccurs="1" maxOccurs="1" />
                                <!-- ipv4address is made optional as there could be virtual contexts
                                    configured without ip -->
                                <xs:element name="deviceName" type="xs:string"
                                    minOccurs="1" maxOccurs="1" />
                                <xs:element name="ipv4Address" type="xs:string"
                                    minOccurs="0" maxOccurs="1" />
                                <xs:element name="sysObjectID" type="xs:string"
                                    minOccurs="1" maxOccurs="1" />
                                <xs:element name="gid" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 126: GetListofDeployableDevices Response XSD

3.2.24 Method DeployConfigByGID

Once the user has policy changes, config can be deployed to the device(s). If user has deploy privileges then he can deploy the config. In this release API supports immediate deploy option. The deployment to IPS devices will be blocked in this release. If IPS device is found in the request then an error will be thrown to the user.

If the deployment options are not specified then we will use the values from admin settings.API will read the deployment options that are set (or default values) using CSM client and will use while deploying to the device.

3.2.24.1 Request

The request takes the device GID and the other deployment options as mentioned below in the example and the XSD.

```
URL:

https://hostname/nbi/configservice/deployConfigByGID

<?xml version="1.0" encoding="UTF-8"?>
<csm:deployConfigRequest xmlns:csm="csm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance>"
    <deviceGID>
      <gid>00000000-0000-0000-0000-004294967326</gid>
    </deviceGID>
    <deploymentOptions>
      <DeployOptions>device</DeployOptions>
    </deploymentOptions>
</csm:deployConfigRequest>
```

Figure 127: DeployConfigByGID request Example

Table 88: DeployConfigByGID XML element and attribute Description

XML Element & Attributes	Definition
deployConfigRequest	An element that mentions the details of the deploy operation
Element List: deviceGID	Unique value that specifies the list of device ID for which the changes to be deployed
Element: deploymentOptions	Deployment options which we can override for this deployment. Options like deploy to file or detect OOB etc
ElementList: deploymentComments	Boolean value – true , optionally user can specify comments.
Element: deployDeviceAttrbs.deployOptions.deployMethod.device	Fixed values to mention if we want to deploy to device or file.
Element: deployDeviceAttrbs.deployOptions.deployMethod.file	Deploy to file.
Element: deployDeviceAttrbs.deployOptions.deployMethod.file.filePath	If file option is selected then filePath should be provided

XML Element & Attributes	Definition
Element: deployDeviceAttrbs.deployOptions.OOBdetectionbehaviour	Describes options to handle oob as over write the changes or cancel deployment or do not check for changes.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:complexType name="deployDeviceAttrs">
        <xs:sequence>
            <xs:element name="deployOptions">
                <xs:complexType>
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="deployMethod" type="ObjectIdentifierList"
                            minOccurs="0" maxOccurs="1" />
                        <xs:choice>
                            <xs:element name="device" type="xs:string" minOccurs="0"
                                maxOccurs="1" fixed="device" />
                            <xs:sequence>
                                <xs:element name="file" type="xs:string" minOccurs="0"
                                    maxOccurs="1" fixed="file" />
                                <xs:element name="filePath" type="xs:string"
                                    minOccurs="0" maxOccurs="1" />
                            </xs:sequence>
                        </xs:choice>
                        <xs:element name="OOBdetectionbehavior" minOccurs="0"
                            maxOccurs="1">
                            <xs:simpleType>
                                <xs:restriction base="xs:token">
                                    <xs:enumeration value="over write changes and show warnings"
/>
                                    <xs:enumeration value="cancel deployment" />
                                    <xs:enumeration value="do not check for changes" />
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>
                    </xs:sequence>

                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

<xs:element name="deployConfigRequest" type="DeployConfigRequest" />
    <xs:complexType name="DeployConfigRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="deviceGID" type="ObjectIdentifierList"
                        minOccurs="1" maxOccurs="1">
                    <xs:element name="deploymentOptions" type="deployDeviceAttrs" minOccurs="0"
maxOccurs="1" />
                        <xs:element name="deploymentComments" type="xs:string"
                        minOccurs="0" maxOccurs="1" />
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 128: DeployConfigByGID Request XSD

3.2.24.2 Response

The response will return the deployment job id and name and also the error list of devices that could not be added to the deployment job because of privilege or invalid device id. There will be another API to query the status of the deployment job. Following gives an example of the response example and XSD.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ns1:deploymentResponse xmlns:ns1="csm">
 <protVersion>1.0</protVersion>
 <status>Deployment Job creation Succeeded</status>
 <deploymentGID>00000000-0000-0000-0000-004294968944</deploymentGID>
 <deploymentJobName>admin123_job_2014-03-25 17:03:56.107</deploymentJobName>
 <devicesInProgress>00000000-0000-0000-0000-004294968099</devicesInProgress>
 <devicesFailureList />
</ns1:deploymentResponse>
```

Figure 129: DeployConfigByGID Response Example

Table 89: DeployConfigByGID XML element and attribute Description

XML Element & Attributes	Definition
Deploymentresponse	An element that mentions the details of the deployment
Element List: status	Deployment job status as success or failure
Element List: deploymentGID	Unique identifier for the deployed job
Element List: deploymentJobName	Unique value that specifies the name
ElementList: devicesInProgress	Number of devices whose deployment are in progress.
ElementList: devicesSuccessList	Number of devices for which the deployment went successfully
ElementList: devicesFailureList	Number of devices the deployment failed with errors
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deploymentResponse" type="DeploymentResponse" />
	<xs:complexType name="DeploymentResponse">
		<xs:complexContent>
			<xs:extension base="BaseReqResp">
				<xs:sequence minOccurs="1" maxOccurs="1">
					<xs:element name="status" type="xs:string"	minOccurs="1" maxOccurs="1" />
					<xs:element name="deploymentGID" type="ObjectIdentifier" maxOccurs="1"
minOccurs="1"></xs:element>
					<xs:element name="deploymentJobName" type="xs:string" minOccurs="1"
maxOccurs="1" />
					<xs:element name="devicesInProgress" type="ObjectIdentifier" minOccurs="1"
maxOccurs="unbounded" />
					<xs:element name="devicesSuccessList" type="DeploymentStatus" minOccurs="0"
maxOccurs="unbounded" />
					<xs:element name="devicesFailureList" type="DeploymentStatus" minOccurs="0"
maxOccurs="unbounded" />
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>

	<xs:complexType name="DeploymentStatusDetails">
		<xs:sequence minOccurs="1" maxOccurs="unbounded">
		   <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1" />
		   <xs:element name="deploymentMsgdetails" type="DeploymentDeviceMsgDetails" minOccurs="0"
maxOccurs="unbounded" />
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="DeploymentStatus">
		<xs:sequence minOccurs="1" maxOccurs="unbounded">
		  <xs:element name="deploymentstatus" type="DeploymentStatusDetails" minOccurs="0"
maxOccurs="unbounded" />
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="DeploymentDeviceMsgDetails">
		<xs:sequence minOccurs="1" maxOccurs="unbounded">
			<xs:element name="Title" type="xs:string" minOccurs="1"
				maxOccurs="1" />
			<xs:element name="severityval" minOccurs="1" maxOccurs="1">
				<xs:simpleType>
					<xs:restriction base="xs:token">
						<xs:enumeration value="Critical" />
						<xs:enumeration value="Warning" />
						<xs:enumeration value="Info" />
					</xs:restriction>
				</xs:simpleType>
			</xs:element>
			<xs:element name="description" type="xs:string" minOccurs="1"
				maxOccurs="1" />
			<xs:element name="action" type="xs:string" minOccurs="0"
				maxOccurs="1" />
		</xs:sequence>
	</xs:complexType>
```

Figure 130: DeployConfigByGID response XSD

3.2.25 Method GetDeployJobStatus

This method is used to get the status of the deployment job.

3.2.25.1 Request

The request takes either the job id or name and returns the status of the deployment job. Following gives an example of the request and its XSD.

```
URL:

https://hostname/nbi/configservice/getdeployJobStatus

<csm:deployJobStatusRequest xmlns:csm="csm">
       <deploymentJobName>admin_job_2014-03-26 13:00:38.28</deploymentJobName>
</csm:deployJobStatusRequest>
```

Figure 131: GetDeployJobStatus Request Example

Table 90: GetDeployJobStatus Request Elements and Attribute Description

XML Element & Attributes	Definition
deployJobStatusRequest	An element that mentions the details of the deployment job
Element List:	Deployment job status as success or failure
Element List: deploymentJobGID	Unique identifier for the deployed job
Element List: deploymentJobName	Unique value that specifies the name
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deployJobStatusRequest" type="DeployJobStatusRequest"/>
      <xs:complexType name="DeployJobStatusRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:choice id="selectReqType">
                          <xs:element name="deploymentJobName" type="xs:string" minOccurs="1"
maxOccurs="1" />
                          <xs:element name="deploymentJobGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                        </xs:choice>
                    </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 132: GetDeployJobStatus Request XSD

3.2.25.2 Response

The response of this is same as the response of DeployConfigByGID. Only the example is given in this section for the XSD details please see the response of DeployConfigByGID.

```
<ns1:deploymentResponse>
<protVersion>1.0</protVersion>
<status>Deployment Job Status Retrieved</status>
<deploymentGID>00000000-0000-0000-0000-042949673573</deploymentGID>
<deploymentJobName>admin_job_2014-03-26 13:00:38.28</deploymentJobName>
<devicesSuccessList>
    <deploymentstatus>
        <deviceGID>00000000-0000-0000-0000-004294967309</deviceGID>
        <deploymentMsgdetails>
            <Title>Interface defined on device does not have a name</Title>
            <severityval>Warning</severityval>
            <description>
            Some of the interfaces defined on device 10.106.164.35 does not have a name defined. Rules bound just to these
interfaces will not be deployed.
            </description>
            <action>Please make sure all interfaces have a valid name.</action>
        </deploymentMsgdetails>
    </deploymentstatus>
</devicesSuccessList>
<devicesFailureList>
    <deploymentstatus>
        <deviceGID>00000000-0000-0000-0000-042949673081</deviceGID>
        <deploymentMsgdetails>
            <Title>Unable to Deploy</Title>
            <severityval>Critical</severityval>
            <description>
            An error response from the device prevented successful completion of this operation. The device provided the following
description: no snmp-server user cisco cisco v3 ERROR: Configuration request for SNMP user cisco failed. Host
traphost.cisco.1.1.1.1.4 references user intended for removal.
            </description>
            <action>
            Please review the deployment transcript for details on the error signaled by the device. Please retry the operation. If the
problem persists, please Select Tools->Security Manager Diagnostics and send the file to the Cisco Technical Assistance Center.
            </action>
        </deploymentMsgdetails>
        <deploymentMsgdetails>
            <Title>Deploy Not Completed</Title>
            <severityval>Critical</severityval>
            <description>This device signaled an error during deployment.</description>
            <action>
            Please review the deployment transcript for details on the error signaled by the device.
            </action>
        </deploymentMsgdetails>
    </deploymentstatus>
</devicesFailureList>
</ns1:deploymentResponse>
```

Figure 133: GetDeployJobStatus Response Example

3.2.26 Method AddPolicyConfigByGID

This method is used to create or add a policy to a device. Please see section Policy for the supported list. The request will take in homogeneous policies in a single request for a device and add it to the device. For example to add a single/multiple firewall rules to a device this method should be used. If the policy is already present this will add entries to the firewall table else it will create a new policy group with the given rules on the device. If a device is assigned a shared policy this method cannot be used but addPolicyConfigByName needs to be used. Policy specific details are under Section 3.3.

3.2.26.1 Request

The request should have the policy data, the device id and csmSessionID. The request can have multiple policy data and each request operates in a single transaction in fail-fast mode. The gid attribute within the policy should not be given in the request.

```
URL:

https://hostname/nbi/configservice/addPolicyConfigByGID

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:setPolicyConfigRequest xmlns:p="csm">
   <csmSessionGID>00000000-0000-0000-0000-008589934739</csmSessionGID>
   <deviceGID>00000000-0000-0000-0000-008589934595</deviceGID>
   <deviceAccessRuleUnifiedFirewallPolicy>
     <name/>
     <type>DeviceAccessRuleUnifiedFirewallPolicy</type>
     <orderId>0</orderId>
     <isMandatoryAggregation>true</isMandatoryAggregation>
     <description> </description>
     <configState>committed</configState>
     <isEnabled>true</isEnabled>
     <direction>In</direction>
     <permit>false</permit>
     <interfaceRoleObjectGIDs>
      <gid>00000000-0000-0000-0000-000000000213</gid>
     </interfaceRoleObjectGIDs>
     <sources>
      <ipData>2.2.2.1</ipData>
     </sources>
     <destinations>
      <networkObjectGIDs>
       <gid>00000000-0000-0000-0000-000000000100</gid>
      </networkObjectGIDs>
     </destinations>
     <services>
      <serviceObjectGIDs>
       <gid>00000000-0000-0000-0000-000000001041</gid>
      </serviceObjectGIDs>
     </services>
     <logOptions>
     <isFirewallLoggingEnabled>true</isFirewallLoggingEnabled>
     <isDefaultLogging>true</isDefaultLogging>
     </logOptions>
     <iosOptions>None</iosOptions>
   </deviceAccessRuleUnifiedFirewallPolicy>
</p:setPolicyConfigRequest>
```

Figure 134: AddPolicyConfigByGID Request URL Example

Table 91:AddPolicyConfigByGID Element and Attribute Description

HTTP/XML Content	Definition
setPolicyConfigRequest	Add policy containing request arguments

HTTP/XML Content	Definition
Element: csmSessionGID	An element that uniquely defines a CSMSession
Element: deviceGID	The GID of the device on which the policy is to be created.
Element: name	The name of the shared policy, if the request is for shared policy instead of device.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="setPolicyConfigRequest" type="SetPolicyConfigRequest" />
    <xs:complexType name="SetPolicyConfigRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                    <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
                    </xs:choice>
                    <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element name="deviceAccessRuleUnifiedFirewallPolicy"
type="DeviceAccessRuleUnifiedFirewallPolicy" minOccurs="0"
                            maxOccurs="500" />
                        <xs:element name="deviceAccessRuleFirewallPolicy"
type="DeviceAccessRuleFirewallPolicy" minOccurs="0" maxOccurs="500" />
                        <xs:element name="firewallACLSettingsPolicy"
type="FirewallACLSettingsPolicy" minOccurs="0" maxOccurs="500">
                        </xs:element>
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 135: AddpolicyConfigByGID request XSD.

3.2.26.2 Response

The response of the addpolicy contains the gid's of the policy that got added and also the status code that determines the success or failure of the whole operation. If failure the reponse also contains the policy due to which it got failed along with the error codes.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:policyConfigResponse xmlns:ns1="csm">
    <statusCode>SUCCESS</statusCode>
    <gid>00000000-0000-0000-0000-008589934703</gid>
    <gid>00000000-0000-0000-0000-008589935436</gid>
 </ns1:policyConfigResponse>
```

Figure 136: AddPolicyConfigByGID Response Example

Table 92: AddPolicyConfigByGID XML Attribute description

HTTP/XML Content	Definition
policyConfigResponse	Policy Config Response containing the response attributes
Element: statusCode	The overall status of the operation.
Element: gid	The gid of the policy.
Element: policy	Element that identifies the policy information
Element: errorInfo	Element giving the error code and the error message.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="policyConfigResponse" type="PolicyConfigResponse"/>
  <xs:complexType name="PolicyConfigResponse">
    <xs:complexContent>
      <xs:extension base="BaseReqResp">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="statusCode" type="OperationStatus" minOccurs="1" maxOccurs="1" />
          <xs:element name="gid" type="ObjectIdentifier" minOccurs="0" maxOccurs="500" />
          <xs:element name="policy" type="BasePolicy" minOccurs="0" maxOccurs="500" />
          <xs:element name="errorInfo" type="BaseError" minOccurs="0" maxOccurs="500" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
    </xs:complexType>

  <xs:simpleType name="OperationStatus">
        <xs:restriction base="xs:token">
            <xs:enumeration value="SUCCESS" />
            <xs:enumeration value="FAILED" />
        </xs:restriction>
    </xs:simpleType>
```

Figure 137: AddPolicyConfigByGID Response XSD

3.2.27 Method AddPolicyConfigByName

This method is used to add a policy to an existing shared policy group. Following gives an example of AddPolicyConfigByName request & response. Please see section Policy for the supported list. Policy specific details are under Section 3.3.

3.2.27.1 Request

The request of addPolicyConfigByName is similar to addPolicyConfigByGID except that instead of the deviceGID the shared policy name needs to be mentioned.

```
URL:

https://hostname/nbi/configservice/addPolicyConfigByName

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:setPolicyConfigRequest xmlns:p="csm">
   <csmSessionGID>00000000-0000-0000-0000-008589934739</csmSessionGID>
   <name>sharedAccessRule1</name>
   <deviceAccessRuleUnifiedFirewallPolicy>
     <name/>
     <type>DeviceAccessRuleUnifiedFirewallPolicy</type>
     <orderId>0</orderId>
     <isMandatoryAggregation>true</isMandatoryAggregation>
     <description> </description>
     <configState>committed</configState>
     <isEnabled>true</isEnabled>
     <direction>In</direction>
     <permit>false</permit>
     <interfaceRoleObjectGIDs>
      <gid>00000000-0000-0000-0000-000000000213</gid>
     </interfaceRoleObjectGIDs>
     <sources>
      <ipData>2.2.2.1</ipData>
     </sources>
     <destinations>
      <networkObjectGIDs>
       <gid>00000000-0000-0000-0000-000000000100</gid>
       </networkObjectGIDs>
     </destinations>
     <services>
      <serviceObjectGIDs>
       <gid>00000000-0000-0000-0000-000000001041</gid>
       </serviceObjectGIDs>
     </services>
     <logOptions>
     <isFirewallLoggingEnabled>true</isFirewallLoggingEnabled>
     <isDefaultLogging>true</isDefaultLogging>
     </logOptions>
     <iosOptions>None</iosOptions>
   </deviceAccessRuleUnifiedFirewallPolicy>
</p:setPolicyConfigRequest>
```

Figure 138: addPolicyConfigByName Request Example

3.2.27.2 Response

The response of addPolicyConfigByName is similar to addPolicyConfigByGID.

3.2.28 Method ModifyPolicyConfigByGID

This method is used to modify a policy assigned to a device. Following gives an example of request and response of this method. Please see section Policy for the supported list. Policy specific details are under Section 3.3.

3.2.28.1 Request

The request of modifyPolicyConfigByGID is similar to addPolicyConfigByGID with gid being mandatory for modify cases. The following gives an example request.

```
URL:

https://hostname/nbi/configservice/modifyPolicyConfigByGID

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:setPolicyConfigRequest xmlns:p="csm">
<csmSessionGID>00000000-0000-0000-0000-077309411427</csmSessionGID>
<deviceGID>00000000-0000-0000-0000-038654705767</deviceGID>
<deviceAccessRuleUnifiedFirewallPolicy>
<gid>77309411336</gid>
<type>DeviceAccessRuleUnifiedFirewallPolicy</type>
<orderId>6</orderId>
<isMandatoryAggregation>true</isMandatoryAggregation>
<description> </description>
<configState>deployed</configState>
<isEnabled>true</isEnabled>
<direction>In</direction>
<permit>false</permit>
<policyName>Local</policyName>
<interfaceRoleObjectGIDs>
<gid>00000000-0000-0000-0000-000000000213</gid>
</interfaceRoleObjectGIDs>
<sources>
<ipData>2.2.2.14</ipData>
</sources>
<destinations>
<networkObjectGIDs>
<gid>00000000-0000-0000-0000-000000000100</gid>
</networkObjectGIDs>
</destinations>
<services>
<serviceObjectGIDs>
<gid>00000000-0000-0000-0000-000000001041</gid>
</serviceObjectGIDs>
</services>
<logOptions>
<isFirewallLoggingEnabled>true</isFirewallLoggingEnabled>
<isDefaultLogging>true</isDefaultLogging>
</logOptions>
</deviceAccessRuleUnifiedFirewallPolicy>
</p:setPolicyConfigRequest>
```

Figure 139: modifyPolicyConfigByGID

3.2.28.2 Response

The response is similar to response of addPolicyConfigByGID.

3.2.29 Method ModifyPolicyConfigByName

This method is used to modify a shared policy. Please see section Policy for the supported list. Following gives an example of the request and response. Policy specific details are under Section 3.3.

3.2.29.1 Request

The request is similar to addPolicyConfigByGID with gid being mandatory and shared policy name instead of the deviceGID. Following gives an example of the request:

```
URL:

https://hostname/nbi/configservice/modifyPolicyConfigByName

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:setPolicyConfigRequest xmlns:p="csm">
   <csmSessionGID>00000000-0000-0000-0000-008589934739</csmSessionGID>
   <name>sharedAccessRule1</name>
   <deviceAccessRuleUnifiedFirewallPolicy>
     <gid>77309411336</gid>
     <type>DeviceAccessRuleUnifiedFirewallPolicy</type>
     <orderId>6</orderId>
     <isMandatoryAggregation>true</isMandatoryAggregation>
     <description> </description>
     <configState>committed</configState>
     <isEnabled>true</isEnabled>
     <direction>In</direction>
     <permit>false</permit>
     <interfaceRoleObjectGIDs>
      <gid>00000000-0000-0000-0000-000000000213</gid>
     </interfaceRoleObjectGIDs>
     <sources>
      <ipData>2.2.2.1</ipData>
     </sources>
     <destinations>
      <networkObjectGIDs>
       <gid>00000000-0000-0000-0000-000000000100</gid>
      </networkObjectGIDs>
     </destinations>
     <services>
      <serviceObjectGIDs>
       <gid>00000000-0000-0000-0000-000000001041</gid>
      </serviceObjectGIDs>
     </services>
     <logOptions>
     <isFirewallLoggingEnabled>true</isFirewallLoggingEnabled>
     <isDefaultLogging>true</isDefaultLogging>
     </logOptions>
     <iosOptions>None</iosOptions>
   </deviceAccessRuleUnifiedFirewallPolicy>
</p:setPolicyConfigRequest>
```

Figure 140: modifyPolicyConfigByName Request Example

3.2.29.2 Response

The response is similar to addPolicyConfigByGID.

3.2.30 Method DeletePolicyConfigByGID

This method is used to delete a policy that is assigned to a device. Please see section Policy for the supported list. The following gives the example and XSD of the request and the response object. Policy specific details are under Section 3.3.

3.2.30.1 Request

The request takes in the policy gid and the type of the policy to be deleted.

```
URL:

https://hostname/nbi/configservice/deletePolicyConfigByGID

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:deletePolicyConfigRequest xmlns:p="csm">
<reqId>1234</reqId>
<csmSessionGID>00000000-0000-0000-0000-073014444091</csmSessionGID>
<deviceGID>00000000-0000-0000-0000-021045141429</deviceGID>
<policyGID>00000000-0000-0000-0000-068719476750</policyGID>
<policyType>DeviceAccessRuleFirewallPolicy</policyType>
</p:deletePolicyConfigRequest>
```

Figure 141: deletePolicyConfigByGID Request Example

Table 93: DeletePolicyConfigByGID Request XML attribute description

HTTP/XML Content	Definition
deletePolicyConfigRequest	Delete policy containing request arguments
Element: csmSessionGID	An element that uniquely defines a CSMSession
Element: deviceGID	The GID of the device on which the policy is to be deleted.
Element: name	The name of the shared policy, if the request is for shared policy instead of device.
Element: policyGID	The GID of the policy to be deleted
Element: policyType	The type of the policy to be deleted
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session

HTTP/XML Content	Definition
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="deletePolicyConfigRequest" type="DeletePolicyConfigRequest" />
      <xs:complexType name="DeletePolicyConfigRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                              <xs:choice minOccurs="1" maxOccurs="1">
                                    <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                                    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
                              </xs:choice>
                              <xs:element name="policyGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"
/>
                              <xs:element name="policyType" type="PolicyConfigType" minOccurs="1"
maxOccurs="1" />
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 142: deletePolicyConfigByGID request XSD

3.2.30.2 Response

The response is similar to the response of addPolicyConfigByGID.

3.2.31 Method DeletePolicyConfigByName

This method is used to delete a shared policy. Please see section Policy for the supported list. The following gives an example of the request and response object. Policy specific details are under Section 3.3.

3.2.31.1 Request

The request is similar to deletePolicyConfigByGID with name element instead of deviceGID. The following gives an example of the request:

```
URL:

https://hostname/nbi/configservice/deletePolicyConfigByName

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:deletePolicyConfigRequest xmlns:p="csm">
<reqId>1234</reqId>
<csmSessionGID>00000000-0000-0000-0000-073014444091</csmSessionGID>
<name>asiaRule</name>
<policyGID>00000000-0000-0000-0000-068719476750</policyGID>
<policyType>DeviceAccessRuleFirewallPolicy</policyType>
</p:deletePolicyConfigRequest>
```

Figure 143: DeletePolicyConfigByName request XSD.

3.2.31.2 Response

The response is similar to the response of addPolicyConfigByGID.

3.2.32 Method ReorderPolicyConfigByGID

This method is used to reorder the rules in CSM assigned to a device. It is supported only for access-rules and unified access-rules. The following gives the request and response object. Policy specific details are under Section 3.3.

3.2.32.1 Request

The request takes the rule ID and the toOrder as input and moves the rule to the destined to orderID.

```
URL:

https://hostname/nbi/configservice/reorderPolicyConfigByGID

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:reorderRulesRequest xmlns:p="csm">
<csmSessionGID>00000000-0000-0000-0000-068719476739</csmSessionGID>
<deviceGID>00000000-0000-0000-0000-021045141429</deviceGID>
<ruleGID>00000000-0000-0000-0000-060129542454</ruleGID>
<policyType>DeviceAccessRuleFirewallPolicy</policyType>
<toOrderId>3</toOrderId>
</p:reorderRulesRequest>
```

Figure 144: reorderPolicyConfigByGID request Example

Table 94: ReorderPolicyConfigByGID request element Description

HTTP/XML Content	Definition
deletePolicyConfigRequest	Delete policy containing request arguments
Element: csmSessionGID	An element that uniquely defines a CSMSession
Element**:** deviceGID	The GID of the device on which the policy is assigned.
Element**:** name	The name of the shared policy, if the request is for shared policy instead of device.
Element: ruleGID	The GID of the rule that uniquely defines it.
Element: policyType	The type of the policy.
Element: toOrderID	The orderID where the rule has to be moved.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="reorderRulesRequest" type="ReorderRulesRequest" />
      <xs:complexType name="ReorderRulesRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="csmSessionGID" type="ObjectIdentifier"
                                    minOccurs="0" maxOccurs="1" />
                          <xs:choice minOccurs="1" maxOccurs="1">
                              <xs:element name="deviceGID" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />
                              <xs:element name="name" type="xs:string"
                        minOccurs="1" maxOccurs="1" />
                              </xs:choice>
                              <xs:element name="ruleGID" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />

                              <xs:element name="policyType" type="PolicyConfigType" minOccurs="0"
maxOccurs="1"/>
                              <xs:element name="toOrderId" type="xs:unsignedInt"
                                    minOccurs="1" maxOccurs="1" />
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 145: reorderPolicyConfigByGID request XSD

3.2.32.2 Response

The response is similar to the response of addPolicyConfigByGID.

3.2.33 Method ReorderPolicyConfigByName

This method is used to reorder the rules in CSM assigned to a shared policy. It is supported only for access-rules and unified access-rules. The following gives the request and response object. Policy specific details are under Section 3.3.

3.2.33.1 Request

The request is similar to reorderPolicyConfigByGID except that for deviceGID the shared policy name needs to be given.

```
URL:

https://hostname/nbi/configservice/reorderPolicyConfigByName

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 3-Sep-2013 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<p:reorderRulesRequest xmlns:p="csm">
<csmSessionGID>00000000-0000-0000-0000-068719476739</csmSessionGID>
<name>usRule</name>
<ruleGID>00000000-0000-0000-0000-060129542454</ruleGID>
<policyType>DeviceAccessRuleFirewallPolicy</policyType>
<toOrderId>3</toOrderId>
</p:reorderRulesRequest>
```

Figure 146: reorderPolicyConfigByGID request example

3.2.33.2 Response

The response is similar to the response of addPolicyConfigByGID.

3.2.34 Method CreateSharedPolicy

This method is used to create a shared policy. Qualified shared policy name should be given in the policyName Tag. If shared policy name should be created under an existing parent policy, then it has to be given in the following format <parent-policy-name>//<New-shared-policy-name>

3.2.34.1 Request

The request takes name of the shared policy along with hierarchy. Following gives an example of the request and its XSD.

```
URL:

https://hostname/nbi/configservice/createSharedPolicy

<?xml version="1.0" encoding="UTF-8"?>
<csm:setSharedPolicyConfigRequest xmlns:csm="csm">
<csmSessionGID>00000000-0000-0000-0000-081604378631</csmSessionGID>
<policyName>Global//India//CTS</policyName>
<policyType>DeviceAccessRuleUnifiedFirewallPolicy</policyType>
</csm:setSharedPolicyConfigRequest>
```

Figure 147: CreateSharedPolicy Request Example

Table 95: CreateSharedPolicy Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: `policyName`	Name of the shared policy with hierarchy.
Element: policyType	The type of the policy to be create.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
    <xs:element name="setSharedPolicyConfigRequest" type="SetSharedPolicyConfigRequest" />
    <xs:complexType name="SetSharedPolicyConfigRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                    <!-- Name of the shared policy with hierarchy -->
                    <xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1" />
                    <!-- rename operation new name of the shared policy -->
                    <xs:element name="policyRename" type="xs:string" minOccurs="0" maxOccurs="1" />
                    <!-- Type of shared policy -->
                    <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1" />
                    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 148: CreateSharedPolicy Request XSD

3.2.34.2 Response

The response of the create shared policy have a statusCode that will descripe the overall status of the operation following with a description.

```
<ns1:sharedPolicyConfigResponse>
<protVersion>1.0</protVersion>
<statusCode>SUCCESS</statusCode>
<description>Created shared policy successfully</description>
</ns1:sharedPolicyConfigResponse>
```

Figure 149: CreateSharedPolicy Response Example

Table 96: CreateSharedPolicy Response Elements and Attribute Description

HTTP/XML Content	Definition
`sharedPolicyConfigResponse`	Returns the shared policy config response
Element: `description`	This descripes the status of the shared policy config response
Element: statusCode	This descripes the overall status of the operation

```
<xs:element name="sharedPolicyConfigResponse" type="SharedPolicyConfigResponse" />
<xs:complexType name="SharedPolicyConfigResponse">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="statusCode" type="OperationStatus" minOccurs="1" maxOccurs="1" />
                <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
                <xs:element name="deviceGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 150: CreateSharedPolicy Response XML Schema

Method specific errors

Code	Description
1107	This error will be returned if policy name is not given
1120	This error will be returned if policy Type given as FirewallACLSettingsPolicy and the policy name mentioned as parent//child hierarchy format
1106	This error will be returned if specified parent policy doesn't exist
1112	This error will be returned if the parent policy already have a child with the given name.
1113	This error will be returned if the shared policy name start with dot(.)

Table 97: CreateSharedPolicy Method Error Codes

3.2.35 Method DeleteSharedPolicy

This method is used to delete the given shared policy. If the shared policy is inherited from a parent policy then the policy name should be given as hierarchy format like parent//child.

3.2.35.1 Request

The request takes name of the shared policy along with hirarchy. Following gives an example of the request and its XSD.

```
URL:

https://hostname/nbi/configservice/deleteSharedPolicy


<?xml version="1.0" encoding="UTF-8"?>
<csm:setSharedPolicyConfigRequest xmlns:csm="csm">
<csmSessionGID>00000000-0000-0000-0000-081604378631</csmSessionGID>
<policyName>Global//India//CTS</policyName>
<policyType>DeviceAccessRuleUnifiedFirewallPolicy</policyType>
</csm:setSharedPolicyConfigRequest>
```

Figure 151: DeleteSharedPolicy Request Example

Table 98: DeleteSharedPolicy Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: `policyName`	Name of the shared policy with hierarchy.
Element: policyType	The type of the policy to be deleted
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
    <xs:element name="setSharedPolicyConfigRequest" type="SetSharedPolicyConfigRequest" />
    <xs:complexType name="SetSharedPolicyConfigRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                    <!-- Name of the shared policy with hirarchy -->
                    <xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1" />
                    <!-- rename operation new name of the shared policy -->
                    <xs:element name="policyRename" type="xs:string" minOccurs="0" maxOccurs="1" />
                    <!-- Type of shared policy -->
                    <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1" />
                    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 152: DeleteSharedPolicy Request XSD

3.2.35.2 Response

The response of the delete shared policy have description that “Deleted shared policy successfully” and the status code will be displayed as SUCCESS if the overall delete operation got access

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:sharedPolicyConfigResponse xmlns:ns1="csm">
  <protVersion>1.0</protVersion>
  <statusCode>SUCCESS</statusCode>
  <description>Deleted shared policy successfully</description>
</ns1:sharedPolicyConfigResponse>
```

Figure 153: deleteSharedPolicy Response Example

DeleteSharedPolicy response schema will be as similar as createSharedPolicy response schema mentioned in Figure 150.

DeleteSharedPolicy response XML elements are as similar as createSharedPolicy response elements mentioned in Table 96.

Method specific error codes

Code	Description
1107	This error will be returned if policy name is not given
1120	This error will be returned if policy Type given as FirewallACLSettingsPolicy and the policy name mentioned as parent//child hierarchy format
1106	This error will be returned if specified parent policy doesn’t exist
1092	This error will be returned if the user is not authorized to modify the policy
1093	This error will be returned if the policy is disabled

Table 99: deleteSharedPolicy Method Error Codes

3.2.36 Method RenameSharedPolicy

This method is used to rename a shared policy. The existing shared policy which needs to be renamed should be given in the policyName tag and the new name for rename should be given in the policyRename tag.

3.2.36.1 Request

The request takes name of the shared policy along with hierarchy and new name. Following gives an example of the request and its XSD.

```
URL:

https://hostname/nbi/configservice/renameSharedPolicy


<?xml version="1.0" encoding="UTF-8"?>
<csm:setSharedPolicyConfigRequest xmlns:csm="csm">
<csmSessionGID>00000000-0000-0000-0000-081604378631</csmSessionGID>
<policyName>Global//India//CTS</policyName>
<<policyRename>TCS</policyRename>
<policyType>DeviceAccessRuleUnifiedFirewallPolicy</policyType>
</csm:setSharedPolicyConfigRequest>
```

Figure 154: RenameSharedPolicy Request Example

Table 100: RenameSharedPolicy Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: `policyName`	Name of the shared policy with hierarchy.
Element: `policyRename`	New Name of the shared policy.
Element: policyType	The type of the policy to be renamed.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
    <xs:element name="setSharedPolicyConfigRequest" type="SetSharedPolicyConfigRequest" />
    <xs:complexType name="SetSharedPolicyConfigRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                    <!-- Name of the shared policy with hirarchy -->
                    <xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1" />
                    <!-- rename operation new name of the shared policy -->
                    <xs:element name="policyRename" type="xs:string" minOccurs="0" maxOccurs="1" />
                    <!-- Type of shared policy -->
                    <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1" />
                    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 155: RenameSharedPolicy Request XSD

3.2.36.2 Response

The response of the rename shared policy have description that “Renamed shared policy successfully”.Status code will describe the overall status of the operation.

```
<?xml version="1.0" encoding="UTF-8"?>
 <ns1:sharedPolicyConfigResponse xmlns:ns1="csm">
 <protVersion>1.0</protVersion>
 <statusCode>SUCCESS</statusCode>
 <description>Shared policy renamed successfully</description>
 </ns1:sharedPolicyConfigResponse>
```

Figure 156: RenameSharedPolicy Response Example

renameSharedPolicy response schema will be as similar as createSharedPolicy response schema mentioned in Figure 150.

renameSharedPolicy response XML elements are as similar as createSharedPolicy response elements mentioned in Table 96.

Method specific error codes

Code	Description
1107	This error will be returned if policy name is not given
1120	This error will be returned if policy Type given as FirewallACLSettingsPolicy and the policy name mentioned as parent//child hierarchy format
1106	This error will be returned if specified parent policy doesn’t exist
1121	This error will be returned if policy Rename have been given with a name starting with dot(.) or name having // in it
1092	This error will be returned if the user is not authorized to modify the policy
1093	This error will be returned if the policy is disabled

Table 101: renameSharedPolicy Method Error Codes

3.2.37 Method InheritSharedPolicy

This method is used to inherit a shared policy from another shared policy. This same method can also be used to uninherit or remove the existing inheritance between the policies by setting the <inherit> tag true or false. If the inherit tag set as true means its inheritance operation and if the inherit tag set as false means its uninheritance operation.

3.2.37.1 Request

The request takes name of the shared policy along with hierarchy and the parent name for inherit. While remove the inherit parent name is not required.. Following example gives an example of the request and its XSD.

```
URL:

https://hostname/nbi/configservice/inheritSharedPolicy


<?xml version="1.0" encoding="UTF-8"?>
<csm:inheritSharedPolicyConfigRequest xmlns:csm="csm">
<csmSessionGID>00000000-0000-0000-0000-296352743429</csmSessionGID>
<policyName>TEST2</policyName>
<parentPolicyName>TEST1</parentPolicyName>
<inherit>true</inherit>
<policyType>DeviceAccessRuleFirewallPolicy</policyType>
</csm:inheritSharedPolicyConfigRequest>
```

Figure 157: InheritSharedPolicy Request Example for inherit

```
URL:

https://hostname/nbi/configservice/inheritSharedPolicy


<?xml version="1.0" encoding="UTF-8"?>
<csm:inheritSharedPolicyConfigRequest xmlns:csm="csm">
<csmSessionGID>00000000-0000-0000-0000-296352743429</csmSessionGID>
<policyName>TEST1//TEST2</policyName>
<inherit>false</inherit>
<policyType>DeviceAccessRuleFirewallPolicy</policyType>
</csm:inheritSharedPolicyConfigRequest>
```

Figure 158: InheritSharedPolicy Request Example for uninherit

Table 102: InheritSharedPolicy Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: `policyName`	Name of the shared policy with hierarchy.
Element: `parentPolicyName`	Name of the parent shared policy with hierarchy.
Element: `inherit`	True indicate inheritance and false indicates un inheritance.

XML Element & Attributes	Definition
Element: policyType	The type of the policy to be create.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="inheritSharedPolicyConfigRequest" type="InheritSharedPolicyConfigRequest" />
      <xs:complexType name="InheritSharedPolicyConfigRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"
/>
                              <!-- Name of the shared policy with hirarchy -->
                              <xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1" />
                              <!-- Name of the parent shared policy with hirarchy -->
                              <xs:element name="parentPolicyName" type="xs:string" minOccurs="0" maxOccurs="1" />
                              <!-- create Inheritance or remove Inheritance based on noInheritance
                                    boolean value -->
                              <xs:element name="inherit" type="xs:boolean" minOccurs="1" maxOccurs="1" />
                              <!-- Type of shared policy -->
                              <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1" />
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

Figure 159: InheritSharedPolicy Request XSD

3.2.37.2 Response

The response of the inherit shared policy have description that " shared policy inherited successfully" for inheritance and "shared policy uninherited successfully" for uninheritance.Status code will descripe the overall status of the operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:sharedPolicyConfigResponse>
<protVersion>1.0</protVersion>
<statusCode>SUCCESS</statusCode>
<description>Shared policy inherited successfully</description>
</ns1:sharedPolicyConfigResponse>
```

Figure 160: InheritSharedPolicy Response Example for Inheritance

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:sharedPolicyConfigResponse>
<protVersion>1.0</protVersion>
<statusCode>SUCCESS</statusCode>
<description>Shared policy uninherited successfully</description>
</ns1:sharedPolicyConfigResponse>
```

Figure 161: InheritSharedPolicy Response Example for UnInheritance

InheritSharedPolicy response schema will be as similar as createSharedPolicy response schema mentioned in Figure 150.

InheritSharedPolicy response XML elements are as similar as createSharedPolicy response elements mentioned in Table 96.

Method specific error codes

Code	Description
1107	This error will be returned if policy name is not given
1120	This error will be returned if policy Type given as FirewallACLSettingsPolicy and the policy name mentioned as parent//child hierarchy format
1108	This error will be returned if a policy name is given to inherit from itself
1109	This error will be returned if we try to inherit the policy from the same parent
1111	This error will be returned if the parent policy already has a child with same name. Two policies with same name cannot inherit from the same policy
1106	This error will be returned if specified parent policy doesn't exist

Table 103: inheritSharedPolicy Method Error Codes

3.2.38 Method AssignSharedPolicy

This method is used to assign the given shared policy to the list of devices provided.List of device GID's should be given in the deviceGID List. Policy Name to be assigned should be given in the policyName tag. During assignment if user wants to overwrite the existing local policies in the devices then inheritLocalPolicy Tag should be configured as false. If user wants to inherit the device local policies with the given shared policy then inheritLocalPolicy Tag option should be configured as true.

3.2.38.1 Request

This request takes name of the shared policy along with hierarchy , policy Type and device GID list. To overwrite the local policies in device `inheritLocalPolicy` Tag should be configured as false

```
URL:

https://hostname/nbi/configservice/assignSharedPolicy

<?xml version="1.0" encoding="UTF-8"?>
<ns1:assignSharedPolicyConfigRequest xmlns:ns1="csm">
<csmSessionGID>00000000-0000-0000-0000-042949672963</csmSessionGID>
<policyName>test-config</policyName>
<policyType>DeviceAccessRuleUnifiedFirewallPolicy</policyType>
<deviceGIDs>
         <gid>00000000-0000-0000-0000-017179869219</gid>
</deviceGIDs>
<inheritLocalPolicy>false</inheritLocalPolicy>
</ns1:assignSharedPolicyConfigRequest>
```

Figure 162: AssignSharedPolicy Request Example

Table 104: AssignSharedPolicy Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: `policyName`	Name of the shared policy with hierarchy.
Element: policyType	The type of the policy to be create.
Element: deviceGIDs	List of devices device GID to be assigned
Element: inheritLocalPolicy	True if local policies needs to be inherited , False if local policies can be overwritten
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="assignSharedPolicyConfigRequest" type="AssignSharedPolicyConfigRequest" />
<xs:complexType name="AssignSharedPolicyConfigRequest">
<xs:complexContent>
<xs:extension base="BaseReqResp">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1" />
<xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1" />
<xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="deviceGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1" />
<xs:element name="inheritLocalPolicy" type="xs:boolean" minOccurs="1" maxOccurs="1" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Figure 163: AssignSharedPolicy Request XSD

3.2.38.2 Response

The response of the assign shared policy have description that "`Assignment done sucessfully`" if assignement got success and following the list of device GID's the assignment done will be displayed. statusCode will descripe the overall operation status .

```
<ns1:sharedPolicyConfigResponse>
<protVersion>1.0</protVersion>
<statusCode>SUCCESS</statusCode>
<description>Assignment done sucessfully</description>
<deviceGIDs>
<gid>00000000-0000-0000-0000-021474836974</gid>
</deviceGIDs>
</ns1:sharedPolicyConfigResponse>
```

Figure 164: AssignSharedPolicy Response Example

assignSharedPolicy response schema will be as similar as createSharedPolicy response schema mentioned in Figure 150.

assignSharedPolicy response XML elements are as similar as createSharedPolicy response elements mentioned in Table 96.

Method specific error codes

Code	Description
1118	This error will be returned if the deviceGID is invalid or no privilege to do assign
1107	This error will be returned if policy name is not given
1116	This error will be returned if the parent for the given shared policy couldn't find
1114	This error will be returned if the root id couldn't get retrieved for inheritence
1115	This error will be returned if given shared policy assignement failed
1106	This error will be returned if specified parent policy doesn't exist

Table 105: assignSharedPolicy Method Error Codes

3.2.39 Method UnAssignSharedPolicy

This method is used to unassign the given shared policy from the list of devices. Shared policy name which is going to be unassign should be given in the policyName tag and the list of devices GID shoud be given in the DeviceGIDS tag.

3.2.39.1 Request

The request takes name of the shared policy along with hierarchy and device GID's. Following gives an example of the request and its XSD.

```
URL:

https://hostname/nbi/configservice/unassignSharedPolicy

<?xml version="1.0" encoding="UTF-8"?>
<ns1:unassignSharedPolicyConfigRequest xmlns:ns1="csm">
<csmSessionGID>00000000-0000-0000-0000-021474837936</csmSessionGID>
<policyName>Test</policyName>
<policyType>DeviceAccessRuleUnifiedFirewallPolicy</policyType>
<deviceGIDs>
         <gid>00000000-0000-0000-0000-021474836974</gid>
</deviceGIDs>
</ns1:unassignSharedPolicyConfigRequest>
```

Figure 165: UnAssignSharedPolicy Request Example

Table 106: UnAssignSharedPolicy Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: `policyName`	Name of the shared policy with hierarchy.
Element: policyType	The type of the policy to be create.
Element: deviceGIDs	List of devices device GID to be assigned
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:element name="unassignSharedPolicyConfigRequest" type="UnAssignSharedPolicyConfigRequest" />
<xs:complexType name="UnAssignSharedPolicyConfigRequest">
<xs:complexContent>
<xs:extension base="BaseReqResp">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1" />
<xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1" />
<xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="deviceGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Figure 166: UnAssignSharedPolicy Request XSD

3.2.39.2 Response

The response of the unassign shared policy have description that “UnAssignment done successfully” and status code descripes the overall status of the operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:sharedPolicyConfigResponse>
<protVersion>1.0</protVersion>
<statusCode>SUCCESS</statusCode>
<description>UnAssignment done sucessfully</description>
<deviceGIDs>
<gid>00000000-0000-0000-0000-021474836974</gid>
</deviceGIDs>
</ns1:sharedPolicyConfigResponse>
```

Figure 167: UnAssignSharedPolicy Response Example

unassignSharedPolicy response schema will be as similar as createSharedPolicy response schema mentioned in Figure 150.

unassignSharedPolicy response XML elements are as similar as createSharedPolicy response elements mentioned in Table 96.

Method specific error codes

Code	Description
1118	This error will be returned if the deviceGID is invalid or no privilege to do assign
1011	This error will be returned if its invalid CSM Session ID
1107	This error will be returned if policy name is not given
1119	This error will be returned if policy Type and policy name mismatch
1117	This error will be returned if unassignmnet got failed

Table 107: unassignSharedPolicy Method Error Codes

3.2.40 Method getPolicyObjectsListByType

This method is used to list available policy objects for the requested policy object type.

3.2.40.1 Request

Following gives an example of the request and its XSD.

```
URL:
https://hostname/nbi/configservice/getPolicyObjectsListByType
<?xml version="1.0" encoding="UTF-8"?>
<csm:policyObjectsListByTypeRequest xmlns:csm="csm">

<policyObjectType>NetworkPolicyObject</policyObjectType>
</csm:policyObjectsListByTypeRequest>
```

Figure 168: getPolicyObjectsListByType Request Example

Table 108: getPolicyObjectsListByType Request Elements and Attribute Description

XML Element & Attributes	Definition
Element**:** csmSessionGID	An element that provides the unique GID of the csmSession.
Element: policyObjectType	The type of the policy object to retrieve the list.
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:complexTypename="PolicyObjectsListByTypeRequest">
        <xs:complexContent>
            <xs:extensionbase="BaseReqResp">
                <xs:sequenceminOccurs="1"maxOccurs="1">
                    <xs:elementname="policyObjectType"type="PolicyObjectType"
                        minOccurs="1"maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 169**: getPolicyObjectsListByType Request XSD**

3.2.40.2 Response

The response is same as Response

The following are error code specific to this method. There could be additional generic errors that the method might return in cases of error

Code	Description
1123	This error will be returned if there is No Policy Object(s) are available in Security Manager server.

Table 109: getPolicyObjectsListByType Method Error Codes

3.3 Policy-Specific Handling

The following section gives the details of policy-specific handling for individual policies, especially in the write use cases. In general, for all local device policies that are not shared, methods ending with GID should be used (e.g., addPolicyConfigByGID and modifiyPolicyConfigByGID). However, if that policy is shared, then these methods cannot be used. Instead, methods ending with Name (e.g., addPolicyConfigByName and deletePolicyConfigByName) need to be used after determining the shared policy name that is assigned to the device. The changes made to the shared policy will affect all the devices that the shared policy is assigned to.

3.3.1 DeviceAccessRuleFirewallPolicy

The details given below are similar for deviceAccessRuleFirewallPolicy and deviceAccessRuleUnifiedFirewallPolicy. In the case of inherited policies on the device, the local policies can be modified by using the add/modify/delete byGIDs methods, but the other policies (from which it is inheriting) need to be modified by using the add/modify/delete byName methods. Because each rule has the policy inheritance information, it is easy to determine which method needs to be used. Also, the GID element should not be present while adding or creating a new access-list entry; for modification and deletion cases, GID is mandatory. To understand the individual elements please see Section 3.1.5.1 and Section 3.1.5.2.

The following points needs to be kept in mind during the write operations:

1. The position to insert the rule is determined by the **orderID** that is given in the request XML. The orderID given in the request must be present in the already-exisiting rule, and the add operation will push the existing rule with the given orderID to a position lower and insert the requested rule in the given position. If no orderID is given (e.g., orderID element is not present), then the rule is inserted in the last position.

2. If the rule needs to be inserted within a **section**, then the name of the section can be given in the request, and if no orderID is given, then the rule will be inserted as the last rule in the section. The rules can also be inserted within a section by giving the proper orderID without giving the section name.

3. In the case of a **bulk request,** it is sufficient for the first rule to have the proper orderID; the remaining rules in the request are added in the subsequent lower positions. Rules cannot be inserted in non-contiguous positions in a single bulk request but can be achieved by using more than one add request.

4. If a requested rule that is a global rule is given an orderID that is higher than an interface-specific rule, then the add method will throw an error message.

5. In a single bulk request, **global** and **interface-**specific rules cannot both be added.

6. In the case of shared policies in a single request, rules cannot be added to both mandatory and default sections.

7. Sections cannot be created or edited via the APIs in Version 2.0.

8. More than one rule rule can be modified/deleted by using a single bulk request.

9. To add a **global access rule,** the gid value under <**interfaceRoleObjectGIDs**> should be **213**.

10. If user-defined names are required for access lists, then FirewallACLSettingsPolicy should be used.

3.3.2 FirewallACLSettingsPolicy

This policy is used mainly for giving user-defined names for the access list. For details of the policy, please see Section 3.1.5.3. To specify a user-defined name for global rules, the value of the element "**interfaceGID**" must be **213**.

4 CSM Events Service API

The events service API provides a mechanism for the client to receive the event notifications from the CSM server based on operations performed at CSM.

The event notifications will cover configuration updates initiated by CSM on the managed devices. Out-of-band (OOB) configuration changes on the managed devices will only be supported for ASA (Version 7.2 and above) if the Health and Performance Monitoring (HPM) feature in CSM is enabled.

The event notifications will also cover Device Status change events. For the Device Status change event notifications to work the Device should be managed and monitered by CSM and HPM respectively.

4.1 Methods

The events service defines the following methods:

1. GetServiceInfo
    a. Returns the service-specific information, including name of service, version, date, etc.
2. EventSubscription
    a. Allows a client to register for event notifications.

4.1.1 Method GetServiceInfo

The GetServiceInfo method returns the service description, version information, and pertinent attributes related to the service. The request, response, and object model are the same as described in Section 3.2.1.

4.1.2 Method EventSubcription

The eventSubscription method allows a CSM client to subscribe for event notifications that are filtered based on the criteria specified by the CSM client in the request.The event subscription API only supports configuration change notification syslog events and Device Status Change notification syslog events.

4.1.2.1 Request

An example of the method eventSubscription request is shown in the figure below. The fields in these messages are described in Table 110.

```
URL:

https://hostname/nbi/eventservice/eventSubscription

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
<eventSubRequest>
      <op>add</op>
      <subscriptionId>123454</subscriptionId>
      <eventFilterItem>
          <filterEventType>syslog</filterEventType>
          <filterEventFormat>xml</filterEventFormat>
          <filterEventCategory>configChange</filterEventCategory>
      </eventFilterItem>
      <syslogServer>
           <port>514</port>
           <destAddress>12.1.1.1</destAddress>
      </syslogServer>
</eventSubRequest>
```

Figure 170: eventSubscription for ConfigChange XML Example

The above request will be sent by the client to register for events. The API only supports event notifications using the syslog protocol (the format of the event data over syslog can be *XML* or syslog *plainText* which is described in the table and subsequent sections).

All event subscriptions are linked to the active logged in session. If the session (that initiated the subscription request) is **logged out or expires**, then all subscriptions corresponding to the session are deleted. It is thus important for the client to keep the **session alive** if it needs to receive event notifications.

The following events are supported:

1. Config Change Event
    a. The event will be sent whenever a configuration change has been made and deployed on the network device identified in the event (either via CSM or out-of-band).
2. Device Status Change Event:
    a. This event will be sent whenever CSM detects a change in the device connectivity status, either UP or DOWN.

Table 110: eventSubscription Request Elements and Attributes Descriptions

Element.Attribute Name	Definition
eventSubRequest	eventSubscription request allows the client to register a events handler for one or more classes
eventSubRequest.op	The operation being performed which is one of the enumerated list { add, delete }. The add operation registers a new subscription for the event handler.op The delete operation deletes the subscription and the CSM client will no longer receive any events for that subscription ID.

Element.Attribute Name	Definition
eventSubRequest.subscriptionId	The subscription identifier that uniquely identifies this subscription registered by the CSM client. The subscription identifier will be returned with each event notification matching the filter specified in the request.
eventSubRequest.eventFilterItem	Object EventFilterItem specifies a filter that identifies a type of events that the CSM client requests notification on.
eventSubRequest.eventFilterItem.filterEventType	In 1.0 this is defined as “syslog”. Future types will be supported post-1.0.
eventSubRequest .eventFilterItem.filterEventCategory	Specifies the event category the client is interested in. Multiple filterEvents can be specified to listen to multiple types of events. configChange and deviceStatus are supported.
eventSubRequest.eventFilterItem.filterEventFormat	The event format that is used in the notification message. This is an enumerated list of formats. The list is as follows { xml, plainText }
eventSubRequest.SyslogServer	Is included in the subscription if the client wishes to receive the events over the syslog protocol at a specific destination and port number. Events that are not of syslog format originally will be encapsulated.
eventSubRequest.syslogServer.port	The UDP port number that the syslog relay will forward the event to
eventSubRequest.syslogServer.destAddress	The IPv4 Address that will receive the forwarded syslog events
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:simpleType name="EventType">
        <xs:restriction base="xs:token">
                <xs:enumeration value="syslog"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="EventFormat">
        <xs:restriction base="xs:token">
                <xs:enumeration value="xml"/>
                <xs:enumeration value="plainText"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="EventCategory">
        <xs:restriction base="xs:token">
                <xs:enumeration value="configChange"/>
                <xs:enumeration value="deviceStatus"/>
        </xs:restriction>
</xs:simpleType>
<xs:complexType name="EventFilterItem">
        <xs:sequence>
                <xs:element name="filterEventType" type="EventType" minOccurs="1" maxOccurs="1"/>
                <xs:element name="filterEventFormat" type="EventFormat" minOccurs="1" maxOccurs="1"/>
                <xs:element name="filterEventCategory" type="EventCategory" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="eventSubRequest" type="EventSubRequest"/>
<xs:complexType name="EventSubRequest">
        <xs:complexContent>
                <xs:extension base="BaseReqResp">
                        <xs:sequence>
                                <xs:element name="op" type="SubscriptionOperation" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="subscriptionId" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="eventFilterItem" type="EventFilterItem" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="syslogServer" type="SyslogServer" minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

Figure 171: EventSubRequest XML Schema

4.1.2.2 Response

An example of the eventSubscription response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version="1.0" encoding="UTF-8"?>
      <eventSubResponse>
         <protVersion>1.0</protVersion>
         <reqId>123</reqId>
         <subscriptionId>12345</subscriptionId>
      </eventSubResponse>
```

Figure 172: eventSubscription Response Example

Table 111: eventSubscription Response Elements and Attributes Description

Element.Attribute Name	Definition
eventSubResponse	Returns whether the subscription was successful or not. If successful it contains the subscriptionId sent by the client.

```
<xs:element name="eventSubResponse" type="EventSubResponse"/>
<xs:complexType name="EventSubResponse">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="subscriptionId" type="xs:string" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 173: EventSubResponse XML Schema

Following are some caveats for the syslog notifications:

- The syslog notifications will only be sent for devices currently managed by CSM i.e. the device must be added to the device inventory and must be "managed" by CSM.
- CSM may optionally deploy changes to Auto Update Server (AUS) and Cisco Networking Services (CNS) Configuration Engine based on how the device is managed in CSM. The AUS/CNS schemes provide an intermediate "staging system" for the configuration updates before these actually get deployed on the real device. Please see http://www.cisco.com/en/US/docs/security/security_management/cisco_security_manager/security_manager/4.4/user/guide/dpman.html#wp938164 for additional details. In cases where the deployment happens to CNS or AUS successfully, the event notification scheme will send a **successful config change** update (It is thus possible that at the time of sending the event, the config change might not be active on the real end device).
- Following are specific points for Out of Band (OOB) notifications:
    - The out-of-band change detection is currently enabled only for ASA devices running ASA 7.2.x and higher). Also, OOB detection of ASA devices is enabled only if these devices are monitored by Health and Performance Monitor (HPM) in CSM.
    - The OOB notifications will only start detecting for OOB changes after at least "one successful event subscription is done". OOB events prior to this registration will be unknown and will not be monitored.
    - The HPM component monitors devices every 5 or 10 minutes based on whether a device is actively or non-actively monitored. Hence OOB events will only be created once the HPM monitoring cycle/poll is completed. I.e It is likely that the OOB event will not be immediately generated as soon as any OOB change is done on a device.
    - The event service detects and notifies all OOB configuration changes including changed CLI/config lines that may not managed by CSM. Detection also includes any CLI that may be changed and reverted back. I.e. the system detects any configuration modified event on the device.
    - As a corner case, an OOB event may not be generated if a deployment via CSM is done immediately succeeding an OOB change (before the HPM module can detect the change). Consider the following case:
        - At time 10:00 AM user does a OOB change on ASA1 that is being monitored every 10 minutes (next monitoring schedule for 10:09 AM)

        - At time 10:03 AM user does a CSM deployment that overwrites the changes on ASA1 and deployment completes successfully
        - At time 10:09 AM, the HPM poll cycle will not recognize the OOB change as it is overwritten by the CSM deployment done at 10:03 AM.
- Following are the specific points for deviceStatus notifications.
    - Notifications will be sent for All the Alerts with description "Device Polling" in Health and Performance Monitoring(HPM) feature of CSM. All the notifications will be sent whenever the CSM server is restarted. Henceforth only Device Status change notifications will be sent.
    - If the Alert description is "Device Polling: Connection Timedout" then a DEVICE_DOWN notification will be sent.
    - If the Alert description is "Device Polling: Connected" then a DEVICE_UP notification will be sent.
    - The HPM component monitors devices every 5 or 10 minutes based on whether a device is actively or non-actively monitored. Hence device status events will only be created once the HPM monitoring cycle/poll is completed. I.e It is likely that the device might have gone down and come up, a device status change notification will not be sent in this case as the HPM would not have detected it.

The following are error codes specific to this method. There could be additional generic errors that the method might return in cases of error

Code	Description
4001	This indicates a duplicate subscription request was made during the event subscription. If two subscriptions are made using the same subscription id (for the same user session) then this error is returned.
4002	This error indicates that the Syslog destination IP address specified was invalid. The IP address must be in the a.b.c.d format without any mask address i.e. something like 192.168.10.10
4003	This error is returned if the user tries to delete a subscription that does not exist.
4004	This error indicates that the syslog port specified was invalid. Valid ports range between 1-65535.
4005	This error indicates an invalid subscriber ID was specified. Subscriber ID's containing only whitespaces are not allowed.
4006	This error indicates that either the "Syslog Server" element or the "Event Filter" element was not specified. Both these elements must be specified for an add request only.
4007	This error indicates that either the "Syslog Server" element or the "Event Filter" element was specified for a delete operation. Both these elements must not be specified for an delete request.

Table 112: EventSubscription Method Error Codes

The following two sections elaborate on the content of syslog notification based on whether an XML based on plain text format was chosen during subscription.

4.1.2.3 Syslog XML Event Notifications

If the client has registered for syslog event forwarding using the SyslogServer option all events will be forwarded to the registered destination IP and port number. The content of the syslog message may be transmitted in XML or PlainText depending on the registration parameter specified by the client. If XML format is chosen the event message will follow the schema as defined below.

Following is an example of a configuration change notification:

```
<n:configChangeEvent xmlns:n="csm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="events.xsd">
    <eventType>syslog</eventType>
    <eventCategory>configChange</eventCategory>
    <time>1697-02-01T00:00:00Z</time>
    <content>this is a syslog event</content>
    <srcIP>12.1.1.1</srcIP>
    <srcGID>00000000-0000-0000-0000-000000000890</srcGID>
    <srcDns>cisco.com</srcDns>
    <srcOSType>asa</srcOSType>
    <deploymentType>Device</deploymentType>
    <updateType>NO_OOB</updateType>
</n:configChangeEvent>
```

Figure 174: Configuration Change Notification Example

The above event notification is sent by the server to the registered syslog listener and is based on the following elements and attributes.

Table 113: ConfigChangeEvent Data Element Descriptions

Element Name	Definition
configChangeEvent	Root element indicating the CSM specific configuration change event notification.
configChangeEvent.eventType	Indicates this message is a syslog message
configChangeEvent.eventCategory	Indicates the event category which is configChange
configChangeEvent.subscriptionId	The original subscription ID used by the client that is now “echoed” back.
configChangeEvent.time	The time this event was generated.
configChangeEvent.content	Any additional message associated with the configuration change.
configChangeEvent.srcIP	The IP address of the device that was changed
configChangeEvent.srcGID	The GID of the device that was changed
configChangeEvent.srcDns	The DNS name of the device that was changed (if any).
configChangeEvent.srcOSType	Indicates the device type ASA, FWSM, IOS, PIX.

Element Name	Definition
configChangeEvent.deploymentType	Indicates the type of deployment (*Not applicable in case of an out of Band change where it is set to "Unknown"*). Allowable values are: • **Device→** Deployment to a device • **File →** Deployment to a file • **AUS →**Deployment to the Auto Update Server • **CNS →** Deployment to Cisco Networking Services, Configuration Engine • **TMS →** Deployment to a Token Management Server
configChangeEvent.updateType	**NO_OOB →** Indicates that this configuration update is not an **Out of Band** (OOB) change i.e. the configuration update to the device was done via Cisco Security Manager. **OOB →** Indicates that this configuration update is an OOB change done outside of Cisco Security Manager. ***CSM can detect OOB changes only for devices running ASA Version 7.2 and higher.***

The following figure shows the schema for those event notifications:

```
    <xs:simpleType name="OSType">
        <xs:restriction base="xs:token">
            <xs:enumeration value="ios"/>
            <xs:enumeration value="fwsm"/>
            <xs:enumeration value="asa"/>
            <xs:enumeration value="ips"/>
            <xs:enumeration value="pix"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="event" type="Event"/>
    <xs:complexType name="Event">
        <xs:choice>
            <xs:element name="configChange" type="ConfigChangeEvent"/>
        </xs:choice>
    </xs:complexType>
    <xs:simpleType name="UpdateType">
        <xs:restriction base="xs:token">
            <xs:enumeration value="NO_OOB"/>
            <xs:enumeration value="OOB"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DeploymentType">
        <xs:restriction base="xs:token">
            <xs:enumeration value="Device"/>
            <xs:enumeration value="File"/>
            <xs:enumeration value="AUS"/>
            <xs:enumeration value="CNS"/>
            <xs:enumeration value="TMS"/>
            <xs:enumeration value="Unknown"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="BaseEventDetails">
        <xs:sequence>
            <xs:element name="eventType" type="EventType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="eventCategory" type="EventCategory" minOccurs="1" maxOccurs="1"/>
            <xs:element name="time" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="content" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DeviceSpecificEvent">
        <xs:complexContent>
            <xs:extension base="BaseEventDetails">
                <xs:sequence>
                    <xs:element name="srcIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="srcGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="srcDns" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="srcOSType" type="OSType" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!-- Config Change Event -->
    <xs:element name="configChangeEvent" type="ConfigChangeEvent"/>
    <xs:complexType name="ConfigChangeEvent">
        <xs:complexContent>
            <xs:extension base="DeviceSpecificEvent">
                <xs:sequence>
                    <xs:element name="deploymentType" type="DeploymentType" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="updateType" type="UpdateType" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
```

Figure 175: Event XML Schema

Following is an example of a Device Status change notification:

```
<n: deviceStatusEvent xmlns:n="csm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="events.xsd">
      < subscriptionId >100</ subscriptionId >
      <eventType>syslog</eventType>
      <eventCategory>deviceStatus</eventCategory>
      <time>2013-01-25T04:47:59.082Z</time>
      <content>Connection Error</content>
      <srcIP>12.1.1.1</srcIP>
      <srcGID>00000000-0000-0000-0000-000000000890</srcGID>
      <srcDns>cisco.com</srcDns>
      <srcOSType>asa</srcOSType>
      <updateType>DEVICE_DOWN </updateType>
</n: deviceStatusEvent>
```

Figure 176: Device Status Notification Example

The above event notification is sent by the server to the registered syslog listener and is based on the following elements and attributes.

Table 114: DeviceStatusEvent Data Element Descriptions

Element Name	Definition
deviceStatusEvent	Root element indicating the Device Status event notification.
deviceStatusEvent.eventType	Indicates this message is a syslog message
deviceStatusEvent.eventCategory	Indicates the event category which is deviceStatus
deviceStatusEvent.subscriptionId	The original subscription ID used by the client that is now "echoed" back.
deviceStatusEvent.time	The time this event was generated.
deviceStatusEvent.content	Any additional message associated with the configuration change.
deviceStatusEvent.srcIP	The IP address of the device that was changed
deviceStatusEvent.srcGID	The GID of the device that was changed
deviceStatusEvent.srcDns	The DNS name of the device that was changed (if any).
deviceStatusEvent.srcOSType	Indicates the device type ASA, FWSM, IOS, PIX.
deviceStatusEvent.updateType	**DEVICE_DOWN** → The Device is not reachable from the CSM Server. **DEVICE_UP** → The Device has come up and is reachable from the CSM Server.

The following figure shows the schema for those event notifications:

```
<xs:simpleType name="OSType">
	<xs:restriction base="xs:token">
		<xs:enumeration value="ios"/>
		<xs:enumeration value="fwsm"/>
		<xs:enumeration value="asa"/>
		<xs:enumeration value="ips"/>
		<xs:enumeration value="pix"/>
	</xs:restriction>
</xs:simpleType>
<xs:element name="event" type="Event"/>
<xs:complexType name="Event">
	<xs:choice>
		<xs:element name="configChange" type="ConfigChangeEvent"/>
	</xs:choice>
</xs:complexType>
<xs:simpleType name="UpdateType">
	<xs:restriction base="xs:token">
		<xs:enumeration value="NO_OOB"/>
		<xs:enumeration value="OOB"/>
		<xs:enumeration value="DEVICE_DOWN"/>
		<xs:enumeration value="DEVICE_UP"/>
	</xs:restriction>
</xs:simpleType>
<xs:simpleType name="DeploymentType">
	<xs:restriction base="xs:token">
		<xs:enumeration value="Device"/>
		<xs:enumeration value="File"/>
		<xs:enumeration value="AUS"/>
		<xs:enumeration value="CNS"/>
		<xs:enumeration value="TMS"/>
		<xs:enumeration value="Unknown"/>
	</xs:restriction>
</xs:simpleType>
<xs:complexType name="BaseEventDetails">
	<xs:sequence>
		<xs:element name="eventType" type="EventType" minOccurs="1" maxOccurs="1"/>
		<xs:element name="eventCategory" type="EventCategory" minOccurs="1" maxOccurs="1"/>
		<xs:element name="time" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
		<xs:element name="content" type="xs:string" minOccurs="1" maxOccurs="1"/>
	</xs:sequence>
</xs:complexType>
<xs:complexType name="DeviceSpecificEvent">
	<xs:complexContent>
		<xs:extension base="BaseEventDetails">
			<xs:sequence>
				<xs:element name="srcIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
				<xs:element name="srcGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
				<xs:element name="srcDns" type="xs:string" minOccurs="0" maxOccurs="1"/>
				<xs:element name="srcOSType" type="OSType" minOccurs="1" maxOccurs="1"/>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
<!—Device Status Event -->
<xs:element name=" deviceStatusEvent " type=" DeviceStatusEvent "/>
<xs:complexType name=" DeviceStatusEvent ">
	<xs:complexContent>
		<xs:extension base="DeviceSpecificEvent">
			<xs:sequence>
				<xs:element name="updateType" type="UpdateType" minOccurs="1" maxOccurs="1"/>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
```

Figure 177: Event XML Schema

4.1.2.4 Syslog PlainText Event Notifications

The notifications in the native format will be sent using the syslog protocol if the registered **filterEventFormat** is specified as **plainText** during event subscription.

The following example shows a configuration change notification:

```
[Mon Aug 29 08:30:21 IST 2011]syslog-configChange-101:10.104.52.71 SUCCEEDED in job
admin_job_2011-08-29 08:30:12.148,10.104.52.71,00000000-0000-0000-0000-017179869189,NO DOMAIN
NAME DEFINED,ios,Device,NO_OOB.
```

The following example shows a Device Status change notification:

```
[Thu Jan 24 20:47:59 PST 2013]syslog-deviceStatus-102:,10.104.52.71,00000000-0000-0000-0000-
042949673287,default.domain.invalid,asa,DEVICE_DOWN
```

The standard format for this message is

```
[time-stamp]<eventType>-<eventCategory>-<subscriptionId>:{Comma separated list of event details –
<content>,<srcIp>,<srcGID>,<srcDNS>,<srcOSType>,<deploymentType>,<updateType>}
```

The order of elements in the comma separated list is in the order of elements in the specific event type defined in the XML schema. If the device does not have an IP address then the string “NO IP DEFINED” (srcIP) is contained in the message and “NO DOMAIN NAME DEFINED” is used if there is no source DNS (srcDNS).

5 CSM Utility Service API

The utility service provides general utilities for access to functions on the network devices. The Utility Service APIs ***always*** fetch the device configuration data directly from the device. The Configuration Service API (defined in Section 3), on the other hand, always fetches the data from the CSM database.

Faster responses and better efficiency is obtained using the Configuration Service API rather than the Utility Service API. This is because the Utility Service API communicates with the device during a method request and there is an additional communication overhead between the CSM application and the network device. This may additionally entail an increased load on the network device.

It is thus recommended to use the Utility Service API under the following circumstances:

- If the corresponding configuration data is not supported by the Configuration ServiceAPI
- If the corresponding configuration data is administratively not managed in CSM. By default all applicable policies are managed in CSM (see the **Policy Management** CSM Administration screen)
- If out-of-band changes are made on the end device (in such cases CSM might not have the updated configuration in its database)

5.1 Object Model

The following objects are defined in the utility service

- DeviceReadOnlyCLICmds
    - Identifies a device and a CLI command to be executed against that device
- DeviceCmdResults
    - Identifies a set of device command results
- DeviceCmdResult
    - Identifies a single device and the command results

```
<xs:simpleType name="Result">
    <xs:restriction base="xs:token">
        <xs:enumeration value="ok"/>
        <xs:enumeration value="timeout"/>
        <xs:enumeration value="failed"/>
    </xs:restriction>
</xs:simpleType>
```

Figure 178: Result XML Schema

5.2 Methods

The utility service defines the following methods:

1. GetServiceInfo
    a. Returns the service-specific information including name of service, version, date, etc.
2. execDeviceReadOnlyCLICmd

a. Executes the identified CLI command on the device specified in the method argument.

5.2.1 Method GetServiceInfo

The GetServiceInfo method returns the service description, version information, and pertinent attributes related to the service. The URL for this request is *https://hostname/nbi/utilservice/GetServiceInfo* .The request, response, and object model are the same as described in Section 3.2.1.

5.2.2 Method execDeviceReadOnlyCLICmds

The execDeviceReadOnlyCLICmds method executes a set of commands against a list identified devices and returns the result of each command back to the CSM Client. This method will use the read-only credentials the CSM server has and therefore may not be used to execute any commands that require credentials beyond read-only for the device. **This method is not implemented for IPS/IDS devices.**

The set of commands that can be executed by this method is read-only commands such as statistics, monitoring commands that provide additional information about the operation of the particular device. For example, a client application may invoke CLI commands to gather statistics from the device via this command. A CSM *view_cli* and *view_device* privilege is necessary to use this method. This method fetches configuration information for one device at a time.

NOTE: If show command output has any W3C XML standard reserved symbols then the API response will encode any such special characters as per W3C standards. For example, the "show version" output from a multi-context device can have output as <context>; then the response will appear with the named character reference as < context>. For more details refer to http://www.w3.org/TR/REC-xml/#syntax

5.2.2.1 Request

An example of the method execDeviceReadOnlyCLICmds request is shown in the figure below. The fields in these messages are described in the table below.

```
URL:

https://hostname/nbi/utilservice/execDeviceReadOnlyCLICmds

HTTP Header:

Set-Cookie: asCookie=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/;
domain=.hostdomain.com

XML Argument:

<?xml version="1.0" encoding="UTF-8"?>
      <execDeviceReadOnlyCLICmdsRequest>
         <protVersion>1.0</protVersion>
         <reqId>123</reqId>
         <deviceReadOnlyCLICmd>
           <deviceIP>12.1.1.1</deviceIP>
           <cmd>show</cmd>
           <argument>run all</argument>
           <execTimeout>5</execTimeout>
         </deviceReadOnlyCLICmd>
      </execDeviceReadOnlyCLICmdsRequest>
```

Figure 179: Method execDeviceReadOnlyCLICmds Request Example

Table 115: Method execDeviceReadOnlyCLICmds Request Elements and Attributes Descriptions

Element.Attribute Name	Definition
execDeviceReadOnlyCLICmdsRequest	The execDeviceReadOnlyCLICmds request executes a command on the set of identified devices

Element.Attribute Name	Definition
deviceReadOnlyCLICmd	The device command details to be executed.The choice of deviceIP; deviceName or deviceGID to identify the device that the command is executed against is given. deviceName and deviceGID will perform more efficiently than deviceIP as they avoid having to search for the device credentials.
deviceIP	The device IP address that the command will be executed against
deviceName	The device Name that the command will be executed against
deviceGID	The device object identifier that the command will be executed against
cmd	Fixed command ”show”. The regex allows mixed case [sS][hH][oO][wW]
argument	The show command arguments. Like “run” to show the running config of the device or “access-list” to show the access list details.
execTimeout	The execTimeout attribute optionally takes timeout in seconds. The execTimeout attribute will take be an unsigned Integer. Default is set to 180 Seconds. CSM will try the exec command for 3 times with a delay of 15 seconds. Each trial will have a timeout of the value provided for the execTimeout attribute.
reqId	The request identifier that uniquely identifies the request/response transaction pair between the CSM client and CSM server
HTTP Method	POST
HTTP Header: **asCookie**	The cookie returned by the login method that identifies the authentication session
Returns	200 OK + XML
	401 Unauthorized

```
<xs:complexType name="DeviceReadOnlyCLICmd">
    <xs:sequence>
        <xs:choice minOccurs="1" maxOccurs="1">
            <xs:element name="deviceIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="deviceName" type="xs:string" minOccurs="1" maxOccurs="1" />
            <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
        </xs:choice>
        <xs:element name="cmd" minOccurs="1" maxOccurs="1">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern value="[sS][hH][oO][wW]"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="argument" type="xs:string" minOccurs="1" maxOccurs="1">
        <xs:element name="execTimeout" type="xs: unsignedInt" minOccurs="0" maxOccurs="1">
    </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:element name="execDeviceReadOnlyCLICmdsRequest"
type="ExecDeviceReadOnlyCLICmdsRequest"/>
<xs:complexType name="ExecDeviceReadOnlyCLICmdsRequest">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence>
                <xs:element name="DeviceReadOnlyCLICmd" type="DeviceReadOnlyCLICmd"
minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 180: ExecDeviceReadOnlyCLICmdsRequest XML Schema

5.2.2.2 Response

An example of the execDeviceReadOnlyCLICmds response is shown in the figure below. The fields in these messages are described in the table below.

```
<?xml version="1.0" encoding="UTF-8"?>
      <execDeviceReadOnlyCLICmdsResponse>
         <protVersion>1.0</protVersion>
         <reqId>123</reqId>
         <deviceCmdResult>
           <deviceIP>12.1.1.1</deviceIP>
           <deviceName>rtr.cisco.com</deviceName>
           <deviceGID>00000000-0000-0000-0000-261993005068</deviceGID>
           <result>ok</result>
           <resultContent>
              FWSM Firewall Version 3.1(16) &lt;context>
              Device Manager Version 5.1(1)
              Compiled on Wed 29-Jul-09 02:10 by fwsmbld
              U27-FWSM up 5 days 3 hours
              Hardware:   WS-SVC-FWM-1, 1024 MB RAM, CPU Pentium III 1000 MHz
              Flash STI Flash 8.0.0 @ 0xc321, 20MB
               0: Int: Not licensed        : irq 5
               1: Int: Not licensed        : irq 7
               2: Int: Not licensed        : irq 11
              Licensed features for this platform:
              Maximum Interfaces          : 100
              Inside Hosts                : Unlimited
              Failover                    : Active/Active
              VPN-DES                     : Enabled
              VPN-3DES-AES                : Enabled
              Cut-through Proxy           : Enabled
              Guards                      : Enabled
              URL Filtering               : Enabled
              Security Contexts           : 250
              GTP/GPRS                    : Disabled
              VPN Peers                   : Unlimited
              Serial Number: SAD11420A0N
              Running Activation Key: 0xe3bbe77c 0x0b82b2ba 0x4014b998 0x6bef38ad
              Configuration has not been modified since last system restart.
           <resultContent>
         </deviceCmdResult>
      </execDeviceReadOnlyCLICmdsResponse>
```

Figure 181: execDeviceReadOnlyCLICmds Response Example

Table 116: execDeviceReadOnlyCLICmds Response Elements and Attributes Description

Element.Attribute Name	Definition
execDeviceReadOnlyCLICmdsResponse	Returns command result
serviceVersion	The service version of the configuration service running
deviceCmdResult	The command result details for the specific device
deviceIP	The device IP address if available.
deviceName	The device Name of the device that returned the result
deviceGID	The device object identifier returned the result
result	The command result enumeration { success, generalFailure, timeout }
resultContent	The command result content is the result was successfully

```
    <xs:complexType name="DeviceCmdResult">
        <xs:sequence>
            <xs:element name="deviceIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
            <xs:element name="deviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="result" type="Result" minOccurs="1" maxOccurs="1"/>
            <xs:element name="resultContent" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>


<xs:element name="execDeviceReadOnlyCLICmdsResponse" type="ExecDeviceReadOnlyCLICmdsResponse"/>
    <xs:complexType name="ExecDeviceReadOnlyCLICmdsResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence>
                    <xs:element name="deviceCmdResult" type="DeviceCmdResult" minOccurs="1"
maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Figure 182: ExecDeviceReadOnlyCLICmdsRequest XML Schema

The following are error codes specific to this method. There could be additional generic errors that the method might return in cases of error

Code	Description
3000	This error will be returned if the API is unable to execute the show commands on the network device.
3001	This error will be returned if CSM Server does not have any device with the requested IP Address.
3002	This error will be returned if the requested device is an IPS Device.
3003	This error will be returned if API encounters any issues when contacting the device, for example send a request for a device with invalid credentials.
27	Exectimeout Failure: No Response from the device within the timeout.

Table 117: ExecDeviceReadOnlyCLICmdsRequest Method Error Codes

6 Error Code and Description

The following section gives all the error codes and it's description.

Code	Description
0	Reserved
1	General Failure
2	Lack of resources
3	Object Creation Failure
4	Authorization Failure: No session found
5	Authorization Failure: Invalid or expired session
6	Internal communication failure
7	Authentication Failure for Invalid username and/or password
8	Authorization Failure: Login session error. The SessionId sent in request is incorrect.
9	The max session limit reached. No more sessions allowed.
10	Connectivity problem : An error was encountered while connecting to a server resource
11	Heartbeat url failure while connecting to a server.
12	An error occured while accessing XML request payload.
13	XML request payload contains no data.
14	Session creation failed.
15	XML request is invalid. (*This error is caused if the XML request does not adhere to the published XML schema or if the XML is illformed or invalid. Though this is a system wide error, this error is usually set inside a specific response object if the application is able to parse the request method that is being called*)
16	Non https protocol used for callback url.
17	protVersion is optional, and if an unsupported version is specified, then this error is returned. The version supported in this release is Version 1.0.
18	Returned if an internal error occurred while fetching config
19	Returned if required input parameters are missing in the request
20	Returned if the requested Policy Type is not correct
21	Internal Error
22	Returned if the user is not authorized to view this policy data.
23	Returned if no configuration data is available for requested input parameters
24	Returned if an internal error encountered by the server (when processing a policy type that is not supported).
25	API Service is disabled

26	API license is not applied
27	Exectimeout Failure: No Response from the device within the timeout
28	Returned if there is already another API request in processing
29	Returned if there is already the same user logged in API.
30	Returned if there is already the same user logged in via pre-bundled CSM thick client
31	The given shared policy name is not valid or not in supported policy type.
1001	This error will be returned if the requested device is not be configured with any of the supported policies.
1002	Returned when device assigned policies retrieval failed
1003	Returned when unable to Retrieve the List of Policies from Policy View
1004	Returned when error occurred during creating or modifying a policy Object
1005	Returned when an invalid device GID is specified in the request
1006	Returned when an invalid policy GID is specified in the request
1007	Returned when an invalid parameter is specified for policy creation
1008	Returned when an activity not in edit state in opened.
1009	Returned when an invalid CSM session ID is passed to create a CSMSession
1011	Returned when an invalid CSM session ID is passed to open or validate or submit csmsession
1012	Returned when a CSMSession could not be validated.
1013	Returned when an invalid policy type is specified in the request
1014	Returned when an invalid ip address is specified in the input
1015	Returned when the policy object name is invalid
1016	Returned when the given policy object name has space inbetween
1017	Returned when the ip address range is not valid
1018	Returned when the given network is not valid
1019	Returned when the network mask is not allowed
1020	Returned when multiple networks are given instead of a single network
1021	Returned when references are given for a non-group object
1022	Returned for an invalid FQDN format
1023	Returned when an FQDN entry is given instead of a group.
1024	Returned when an invalid reference id is given in the input.
1025	Returned when trying to make changes to a policy that is locked.
1026	Returned when an unavailable policy object is referenced
1027	Returned when referencing an unmanaged policy
1028	Returned when the requested building block is not available in CSM

1029	Returned if a overridden object is created before creating the parent object
1030	Returned when circular references are found in the input
1031	Returned when an invalid subtype is given in the input
1032	Error thrown when rule ID is not present in the ACL request
1033	Returned when the policy types are not matching in the request
1034	Returned when a attribute of an input is invalid
1035	Returned when the given id is not of the given type in the request
1036	Returned when a mandatory attribute is missing in the request.
1037	Returned when the policy processing failed due to some internal error
1038	Returned when the policy conversion failed due to some internal error
1039	Returned when the policy is locked
1040	Returned when the policy got failed to be persisted in the database.
1041	Returned when an invalid rule id is present in the ACL request
1043	Error thrown if an global rule is given before an interface specific rule
1044	Error thrown when an interface specific rule is inserted after the global rule
1045	Returned when the rule order ID has changed and global rule has come above an interface specific rule
1046	Returned when the given rule id is duplicate
1047	Returned when an invalid rule order id is given
1048	Returned when an invalid parent id is given for the policy object
1049	Returned when trying to override a overridden policy object
1050	Returned when trying to override a policy object whose isProperty field is false
1051	Returned when isProperty field is tried to be set as false for a BB that has overrides
1052	Returned when trying to override a overridden policy Object
1053	Returned when subtype is not valid for the given policy object
1054	Returned when trying to create a Group for an policy object that doesn’t support it.
1055	Returned when overrides are created on an unsupported policy object
1056	Returned when a network policy object group is created for sub types NR,NF,NH and NN
1057	Returned when a service policy object group is created for sub type SO.
1058	Returned when an invalid protocol is specified in the input
1059	Returned for an invalid or null policy
1060	Returned when an ports are entered for protocols other than TCP/UDP
1061	Returned when port range has invalid entries using the “not equal” operator
1062	Returned when an invalid port is given in the request

1063	Returned when multiple protocols are given where only single protocol is allowed
1064	Returned when invalid date is specified in the request
1065	Returned when a duplicate Firewall Setting is given in the input.
1066	Returned when order ID is modified via modification request. Need to use a reorder API for this case
1067	Returned when a mandatory or default ACL rule is changed to the other.
1068	Returned when an invalid rule GID is specified.
1069	Returned when an invalid parameter is specified for modification policy
1070	Returned when an invalid section name is given in the input.
1071	Returned when trying to modify a policy objects subtype
1072	Returned when heterogenous policy is given in the bulk input
1073	Returned when trying to reorder to the same position again
1074	Returned when trying to reorder beyond a section
1076	Returned when invalid parameters are specified for creation of a new policy.
1077	This error is thrown when the server is unable to retrieve the policies from the database
1078	This error is thrown when the policy input is invalid
1079	Returned when discarding a CSMSession failed
1080	Returned when submitting a CSMSession failed
1081	Returned when opening a CSMSession failed
1082	Returned when creating a CSMSession failed.
1083	Returned when user has insufficient privilege to create or modify a policy object
1084	Returned when trying to delete a policy object that is in-use
1085	Returned when unable to delete a policy object
1086	Returned when auto-approve of the CSMSession failed
1087	Returned when approving a CSMSession failed.
1088	Returned when a user has insufficient privilege.
1089	Returned when trying to approve in non-workflow mode.
1090	Returned when the activity not in EDIT_IN_USE mode, and any modifications are tried using that activity.
1091	Returned when an unauthorized user is trying to use a CSMSession
1092	Returned when an unauthorized user is trying to access a policy
1093	Returned when the policy is disabled
1094	Returned when an invalid deployment name or job GID is given
1095	Returned when an invalid order id is given
1096	Returned when trying to reorder inherited rule

1097	Returned when an shared policy is not modified using API names ending with byName
1098	Returned when isMandatory field is set to false for local rules
1099	Returned when a modification of a firewall name setting policy creates a duplicate
1100	Returned when invalid input is given for firewall name settings policy
1101	Returned when global interface option is given with mandatory fields as empty
1102	Returned when global rules are defined for outside direction
1103	Returned when global interface is configured for ASA device lesser than 8.3
1104	Returned when identity user groups are configured for devices lesser that ASA 8.4(2)
1105	Returned when approve activity API is called when that option is disabled from CSM Admin settings.
2000	This error will be returned if the API is unable to retrieve the list of authorized devices for the user who is logged in.
2001	This error will be returned if API is not able to get the available groups from CSM Server for further processing.
2002	This error will be returned if API is not able to retrieve devices from CSM Server.
2003	This error will be returned if API is not able to find the config state of the device.
2004	This error will be returned if API is not able to retrieve interface details for a device.
2005, 2006	These errors will be returned if the API encounters an internal error when processing device specific data.
2007	This error will be returned if the requested Device GID does not exist in the CSM Server.
2008	This error will be returned if API is not able to communicate with the configuration archive module.
2009	This error will be returned if API is not able to fetch the configuration from the Configuration Archive Module
2010	This error will be returned if the requested device name does not exist in the CSM Server.
2011	This error will be returned if the user is not authorized to view the configuration.
2012	This error will be returned if device name is null or empty in the request.
2013	This error will be returned if API is not able to fetch the internal policy list from the CSM server.
2014	This error is returned when deployement to IPS device is initiated
2015	Returned when the user doesn’t have sufficient privilege on the device.
2016	Returned when the deployment has failed
2017	Returned when there are no dirty device available for deployment
3000	This error will be returned if the API is unable to execute the show commands on the network device.

3001	This error will be returned if CSM Server does not have any device with the requested IP Address.
3002	This error will be returned if the requested device is an IPS Device.
3003	This error will be returned if API encounters any issues when contacting the device, for example send a request for a device with invalid credentials.
4001	This indicates a duplicate subscription request was made during the event subscription. If two subscriptions are made using the same subscription id (for the same user session) then this error is returned.
4002	This error indicates that the Syslog destination IP address specified was invalid. The IP address must be in the a.b.c.d format without any mask address i.e. something like 192.168.10.10
4003	This error is returned if the user tries to delete a subscription that does not exist.
4004	This error indicates that the syslog port specified was invalid. Valid ports range between 1-65535.
4005	This error indicates an invalid subscriber ID was specified. Subscriber ID's containing only whitespaces are not allowed.
4006	This error indicates that either the "Syslog Server" element or the "Event Filter" element was not specified. Both these elements must be specified for an add request only.
4007	This error indicates that either the "Syslog Server" element or the "Event Filter" element was specified for a delete operation. Both these elements must not be specified for an delete request.
4008	This error is returned when event filter item and syslog server are empty for an add request.
4009	This error is returned when event filter item and syslog server are specified for a delete request
4010	This error is returned when the subscription ID is already in use by this user

7 API Scaling

The API will support a variety of deployments. The following guidelines should be followed.

1) There is no hard limit on the number of devices supported by the API. The current CSM recommendation is to have approximately 500 devices per server with event management and other features enabled.

8 CSM Client Protocol State Machine

8.1.1 Overview

There are two pre-requisites before a CSM Client can make use of the services on the CSM Server.

The 1st prerequisite is that the client authenticates to the CSM Server. The 2nd prerequisite is that the client verifies that all necessary services are active on the CSM server and the versions of those services match the version expected on the client. This process will return an authentication cookie that the client must use in subsequent calls to the interface. This flow is shown in the figure below.

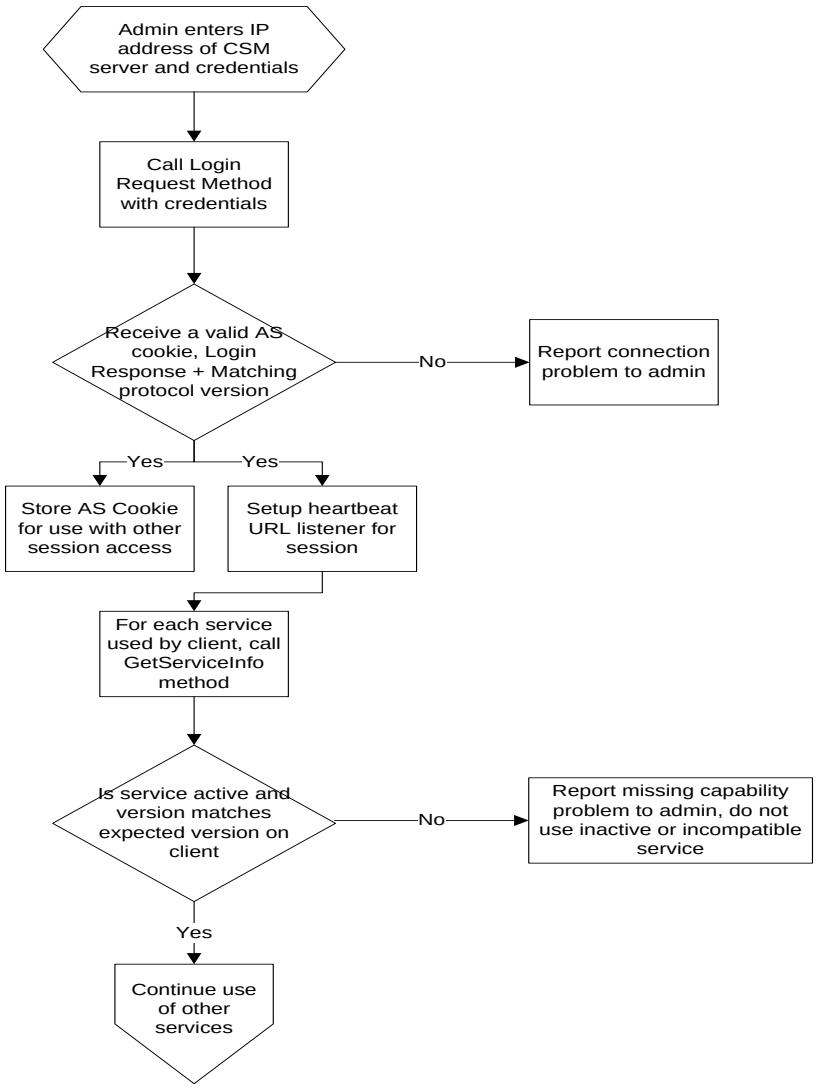


Figure 183: Client Session Initiation Flowchart

Once a session has been established the client may access the service methods for config, events and utility services.

A client that has registered for heartbeat callback should follow the flow defined below.

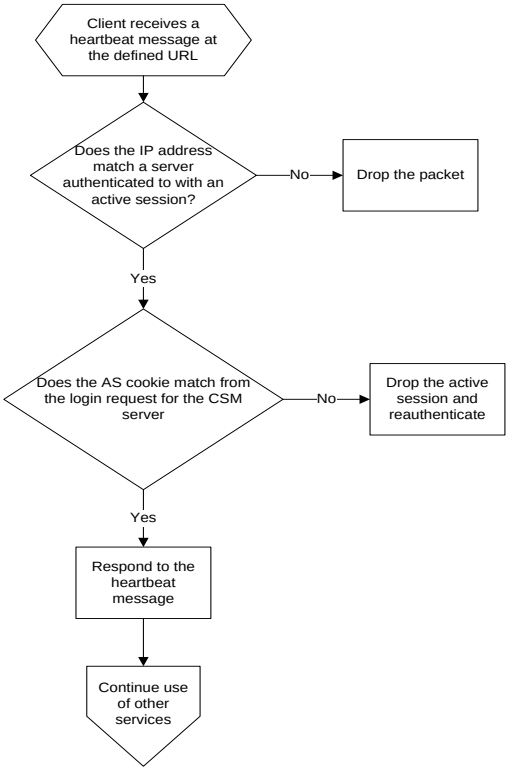


Figure 184: Client Heartbeat Processing Flowchart

A client that has not registered for heartbeat callback but wishes to maintain an active session on the CSM server must call the ping method periodically.

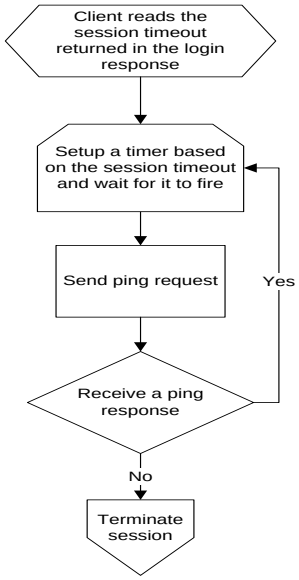


Figure 185: Client Ping Processing Flowchart

8.1.2 Using the configuration and event service

The CSM client may access the API to read configuration of the devices supported by the CSM server. The client may also register for change notifications whenever a configuration change occurs. The following flowchart shows the high-level processing on the client.

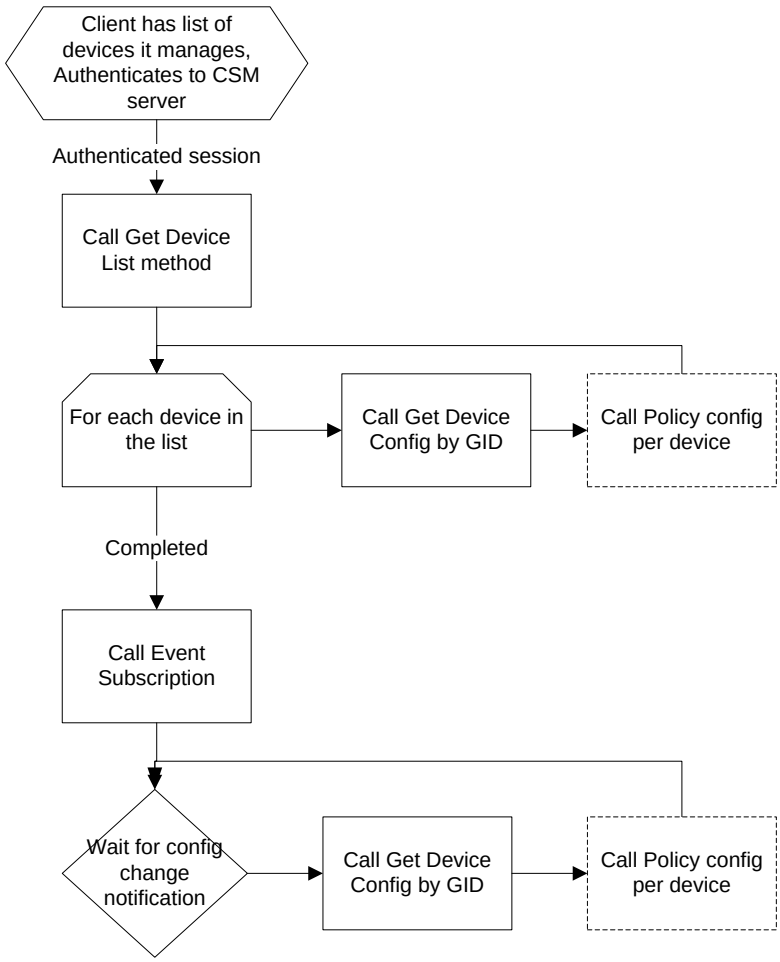


Figure 186: Client Config Flowchart

Following is a typical flow to process policy data in **raw format:**

a. Login (and setup a background thread or process to keep the session alive via a ping or heartbeat)
b. Use one of GetGroupList (or) GetDeviceListByCapability (or) GetDeviceListByGroup to get the list of devices in the system
c. For each of the devices in the list:
    i. Call GetDeviceConfigByGID (or) GetDeviceConfigByName to get the raw config data
d. Logout

Client would additionally need to subscribe to events (see Event Service API in this document), to make the client refresh the configuration if the configuration gets updated. The GetDeviceConfigByGID and GetDeviceConfigByName only return data that is archived inside CSM database. Any updates to the device outside of CSM (out of band) is not available and can only be fetched using the Utility Service API.

Following is a typical flow to process policy data in a policy object model **structured format**:

a. Login (and setup a background thread or process to keep the session alive via a ping or heartbeat)
b. Use one of GetGroupList (or) GetDeviceListBy Capability (or) GetDeviceListByGroup to get the list of devices in the system
c. For each of the devices in the list:
    i. Call GetPolicyListByDeviceGID to find out what "policy types" are configured/supported on a device
    ii. For each of these "Policy Types" call the GetPolicyConfigByDeviceGID or GetPolicyConfigByName (for named/shared policies)
d. Logout

Client would additionally need to subscribe to events (see Event Service API in this document), to make the client refresh the configuration if the configuration gets updated. The GetDeviceConfigByGID and GetDeviceConfigByName only return data that is archived inside CSM database. Any updates to the device outside of CSM (out of band) is not available and can only be fetched using the Utility Service API.

8.1.3 Using CSMSession and Write APIs

For all write APIs, it is mandatory to have a CSMSession in all CSM server modes. The following flowchart describes the flow that needs to be followed:

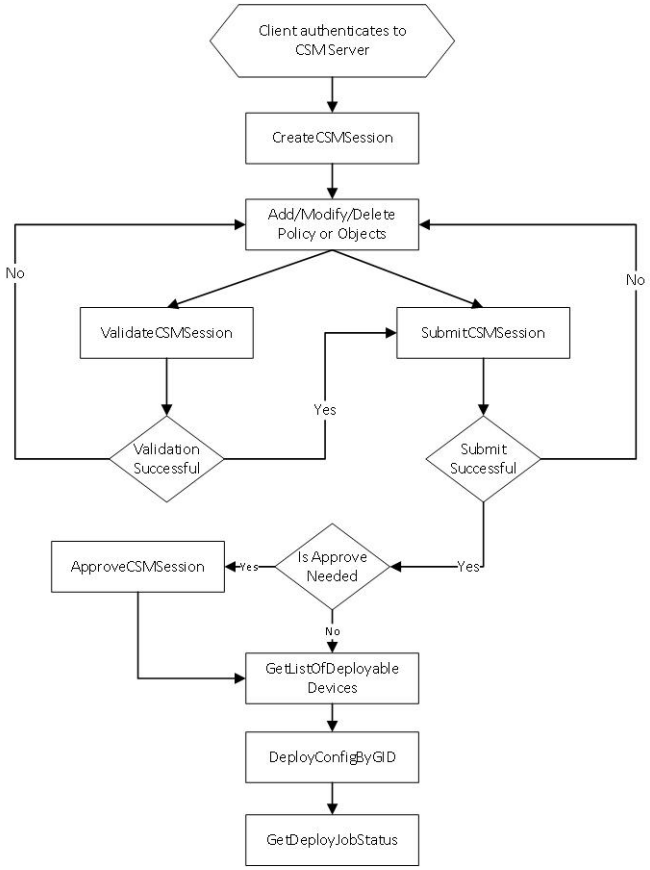


Figure 187: Basic CSMSession Usage

The first step in working with write APIs is to have a CSMSession created or opened. In this flow, we are creating a new CSMSession. Then the respective write APIs are invoked by passing the unique csmSessionID that createCSMSession returns. To get the device ID to be used in the APIs, getDeviceListByCapability() can be used. The next step is to validate the changes for correctness. Because submitCSMSession performs internal validation, it is not mandatory to do validateCSMSession every time. The submitCSMSession based on the CSM server mode returns whether approve is needed or not (that is, when workflow is enabled, approval is required). If approval is needed, then ApproveCSMSession needs to be invoked and then followed by the deployment-related APIs.

In the next flowchart, the use of open and close CSMSessions is explained. CloseCSMSession is used when partial changes are done during a https session, then later at a time when more changes need to be done as part of the same CSMSession, openCSMSession needs to be used. If the changes done during CSMSession need to be reverted, then discardCSMSession should be used.

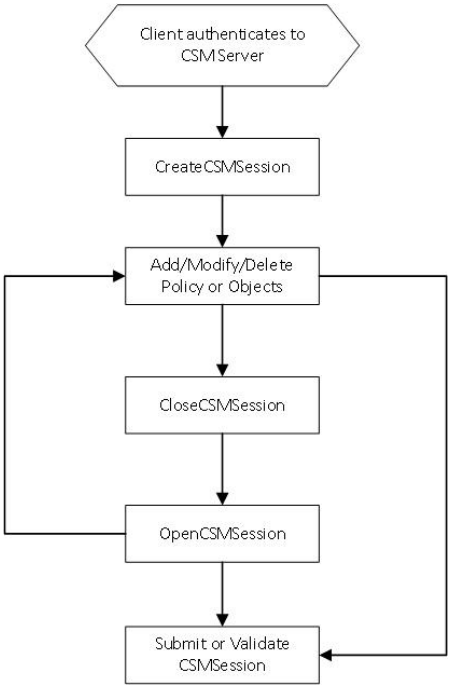


Figure 188: Close and Open CSMSession Usage

9 Sample API Client Programs

NOTE: All the sample programs in this section are provided for simple demonstration purposes only. The programs may need to be appropriately enhanced or modified for use in production systems.

Running the java sample programs mentioned below need the following to be setup:

1. The **http-client** and **http-core** jars from Apache-commons http://hc.apache.org must be included in the java CLASSPATH.
2. The commons-logging jar from Apache-commons http://commons.apache.org/logging/index.html must be included in the java CLASSPATH
3. Define a **CLASSPATH** environment variable in your command shell include all the required jars and other folders as necessary
4. A **client.properties** file must be defined and passed in as a parameter as a java program argument. The format of the file is as follows:

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:pingRequest  xmlns:csm=\"csm\">\
        <protVersion>1.0</protVersion>\
        <reqId>3</reqId>\
</csm:pingRequest>

# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/ping
```

The property definition are as follows:

- **USER**: Defines the username of the user logging in.
- **PASSWORD**: The user password
- **HOST**: The host server to connect
- **XML_REQUEST**: The XML request that needs to be sent (ping in this case)
- **LOGIN_REQUIRED**: If true, a login is done before sending the XML_REQUEST.
- **URI**: Optional URI parameter that indicates the service that needs to be invoked. If this is not specified in the property file, it must be passed in as a java program argument.

Please note that the property values in the client.properties file like HOST, USER, PASSWORD, other values in the XML_REQUEST and the URI itself may need to modified to suit your deployment and the sample that is being executed. Edit this file appropriately before executing any of programs mentioned below.

9.1 CSM API pre-configuration checks

The following sample program implemented in java demonstrates a REST client program that checks if the CSM API is enabled for use. Use the client.properties file as defined above.

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
  * Sample Program to test if CSM server is correctly configured for API
 */
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import java.io.FileInputStream;
import java.util.Properties;

public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
```

```
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        DocumentBuilder domp = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = domp.parse(new ByteArrayInputStream(response.getBytes()));

        NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("code");
        for (int i = 0; i < errorNodes.getLength(); i++) {
            Element element = (Element) errorNodes.item(i);
```

```
            if(element.getTextContent() != null && !element.getTextContent().equals("")) {
                NodeList nodes = doc.getDocumentElement().getElementsByTagName("description");
                for (int j = 0; j < nodes.getLength(); j++) {
                    Element element2 = (Element) nodes.item(j);
                    throw new Exception(element2.getTextContent());
                }
            }
        }
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
```

```
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

9.2 Login and ping test

The following simple sample program implemented in java demonstrates a REST client that logs in to the CSM server using the CSM API and then makes a “ping” request. Use the client.properties file as defined earlier.

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
  * Sample Program to login to the CSM Server and send a ping request
 */
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.ParseException;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;

import java.io.FileInputStream;
import java.util.Properties;

public class RestClient {
```

```
    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params,
schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param PayLoad
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
ClientProtocolException, IOException {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);
```

```
    }

    public void processResponse (HttpResponse httpresponse) throws ParseException, IOException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);

        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
            }
```

```
        }catch(Exception ex){ex.printStackTrace(); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

9.3 Fetch CLI configuration of a firewall

The following simple sample program implemented in java demonstrates a REST client program that uses the CSM API to fetch the raw CLI configuration of a firewall from CSM's database. Use the following client.properties as input (change the name of the device to match the device name in the server's inventory) :

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"utf-8\"?>\
<n:deviceConfigByNameRequest xmlns:n=\"csm\"  xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">\
  <protVersion>1.0</protVersion>\
  <reqId>123</reqId>\
  <name>firewall_device</name>\
</n:deviceConfigByNameRequest>

# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/configservice/getDeviceConfigByName
```

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
 * Sample Program to get entire CLI of a firewall
 */
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
```

```
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import java.io.FileInputStream;
import java.util.Properties;

public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
```

```
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        DocumentBuilder domp = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = domp.parse(new ByteArrayInputStream(response.getBytes()));

        NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("code");
        for (int i = 0; i < errorNodes.getLength(); i++) {
            Element element = (Element) errorNodes.item(i);
            if(element.getTextContent() != null && !element.getTextContent().equals("")) {
                NodeList nodes = doc.getDocumentElement().getElementsByTagName("description");
                for (int j = 0; j < nodes.getLength(); j++) {
                    Element element2 = (Element) nodes.item(j);
                    throw new Exception(element2.getTextContent());
                }
            }
        }
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
```

```
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

9.4 Executing show access-list on a firewall device

The following simple sample program implemented in java demonstrates a REST client using the CSM API to execute a show access-list command on a firewall device. Use the following client.properties as input (change the deviceIP to match the IP of a valid device in the server's inventory) :

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:execDeviceReadOnlyCLICmdsRequest  xmlns:csm=\"csm\">\
  <protVersion>1.0</protVersion>\
  <reqId>123</reqId>\
  <deviceReadOnlyCLICmd>\
    <deviceIP>192.168.1.1</deviceIP>\
    <cmd>show</cmd>\
    <argument>access-list</argument>\
```

```
  </deviceReadOnlyCLICmd>\
</csm:execDeviceReadOnlyCLICmdsRequest>

# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/utilservice/execDeviceReadOnlyCLICmds
```

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
 * Sample program to execute a show access-list command on a firewall
 */
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import java.io.FileInputStream;
import java.util.Properties;

public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }
```

```
    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        DocumentBuilder domp = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = domp.parse(new ByteArrayInputStream(response.getBytes()));
```

```
        NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("code");
        for (int i = 0; i < errorNodes.getLength(); i++) {
            Element element = (Element) errorNodes.item(i);
            if(element.getTextContent() != null && !element.getTextContent().equals("")) {
                NodeList nodes = doc.getDocumentElement().getElementsByTagName("description");
                for (int j = 0; j < nodes.getLength(); j++) {
                    Element element2 = (Element) nodes.item(j);
                    throw new Exception(element2.getTextContent());
                }
            }
        }
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
```

```
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
     }
}
```

9.5 Fetch CSM defined firewall policy

The following simple sample program implemented in java demonstrates a REST client that fetches the CSM firewall policy as it is defined in the CSM UI. Use the following client.properties file (Change the gid value to match the GID of a device in the server's inventory) :

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:policyConfigByDeviceGIDRequest  xmlns:csm=\"csm\">\
  <protVersion>1.0</protVersion>\
  <reqId>123</reqId>\
  <gid>00000000-0000-0000-0000-004294967307</gid>\
 <policyType>DeviceAccessRuleFirewallPolicy</policyType>\
</csm:policyConfigByDeviceGIDRequest>

# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/configservice/getPolicyConfigById
```

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
 * Sample Program to get access rules defined on a firewall as it appears in the
 * CSM UI.
 */
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;
```

```
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import java.io.FileInputStream;
import java.util.Properties;

public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
```

```
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        DocumentBuilder domp = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = domp.parse(new ByteArrayInputStream(response.getBytes()));

        NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("code");
        for (int i = 0; i < errorNodes.getLength(); i++) {
            Element element = (Element) errorNodes.item(i);
            if(element.getTextContent() != null && !element.getTextContent().equals("")) {
                NodeList nodes = doc.getDocumentElement().getElementsByTagName("description");
                for (int j = 0; j < nodes.getLength(); j++) {
                    Element element2 = (Element) nodes.item(j);
                    throw new Exception(element2.getTextContent());
                }
            }
        }
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
```

```
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

9.6 List shared policies assigned to all devices

The following simple sample program implemented in java demonstrates a REST client that iterates over CSM's device inventory and lists the directly assigned shared policies on all devices. Use the following client.properties file:

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:deviceListByCapabilityRequest xmlns:csm=\"csm\">\
  <protVersion>1.0</protVersion>\
  <reqId>123</reqId>\
  <deviceCapability>*</deviceCapability>\
</csm:deviceListByCapabilityRequest>
XML_REQUEST1=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:policyListByDeviceGIDRequest xmlns:csm=\"csm\">\
  <protVersion>1.0</protVersion>\
  <reqId>123</reqId>\
  <gid>DEVICE_ID</gid>\
</csm:policyListByDeviceGIDRequest>

# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/configservice/getDeviceListByType
URI1=https://localhost/nbi/configservice/getPolicyListByDeviceGID
```

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
 * Sample program to collect the Shared Access Rules applied to Devices in CSM.
 */
import java.io.*;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
```

```
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import java.util.ArrayList;
import java.util.Properties;

public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
```

```
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");

                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            String policy_req = prop.getProperty("XML_REQUEST1");
            String policy_req_path = prop.getProperty("URI1");

            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
```

```
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2: Get All Devices
                ArrayList devices = client.getDeviceList(uri, payload, host, true);

                //Step3: Get Shared Policies on Devices.
                ArrayList sharedAccessRules = new ArrayList();
                for (int i = 0; i < devices.size(); i++) {
                    String device = (String) devices.get(i);
                    String policy_request = policy_req.replace("DEVICE_ID", device);
                    ArrayList policies = client.getPolicyList(new URI(policy_req_path),
policy_request, host, true);
                    if(policies!= null) {
                        sharedAccessRules.addAll(policies);
                    }
                    System.out.println(sharedAccessRules);
                }
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    private ArrayList getPolicyList(URI uri, String payload, String host, boolean isCookieNeeded)
throws Exception {
        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        //processResponse(httpresponse);
        return getPolicyList(httpresponse);
    }

    private ArrayList getPolicyList(HttpResponse httpresponse) throws IOException, SAXException,
ParserConfigurationException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        System.out.println(response);
        ArrayList<String> retArr = new ArrayList();
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    DocumentBuilder domp =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
                            Document doc = domp.parse(new
ByteArrayInputStream(response.getBytes()));

                    NodeList errorNodes =
doc.getDocumentElement().getElementsByTagName("policyDesc");
                    for (int i = 0; i < errorNodes.getLength(); i++) {
                        Element element = (Element) errorNodes.item(i);
                        String text = element.getTextContent();
```

```
                        if(text != null && !text.equals("")&&
text.indexOf("DeviceAccessRuleFirewallPolicy")>=0) {
                            text = text.replaceAll("DeviceAccessRuleFirewallPolicy", "");
                            if(!text.equals("-- local --"))// Filter out Local rules.
                                retArr.add(text);
                        }
                    }

                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
        return retArr;
    }

    private ArrayList getDeviceList(URI uri, String payload, String host, boolean isCookieNeeded)
throws Exception {
        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        //processResponse(httpresponse);
        return getDeviceList(httpresponse);
    }

    private ArrayList getDeviceList(HttpResponse httpresponse) throws IOException, SAXException,
ParserConfigurationException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        ArrayList<String> retArr = new ArrayList();
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    DocumentBuilder domp =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
                            Document doc = domp.parse(new
ByteArrayInputStream(response.getBytes()));

                    NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("gid");
                    for (int i = 0; i < errorNodes.getLength(); i++) {
                        Element element = (Element) errorNodes.item(i);
                        if(element.getTextContent() != null &&
!element.getTextContent().equals("")) {
                            retArr.add(element.getTextContent());
                        }
                    }

                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
```

```
                    //Do something...
                }
            }
        }
        return retArr;
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

9.7 List content of a given shared policy

The following simple sample program implemented in java demonstrates a REST client the lists the content of a firewall access rule shared policy given the name of the shared policy. Use the client.properties file as defined below (change shared policy name defined in the properties file appropriately) :

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:policyConfigByNameRequest  xmlns:csm=\"csm\">\
  <protVersion>1.0</protVersion>\
  <reqId>123</reqId>\
  <name>ACL1</name>\
  <policyType>DeviceAccessRuleFirewallPolicy</policyType>\
</csm:policyConfigByNameRequest>
# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/configservice/getPolicyConfigByName
```

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
 * Sample program to get the contents of a shared policy given the shared policy name.
 */
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
```

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import java.io.FileInputStream;
import java.util.Properties;

public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
```

```
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        DocumentBuilder domp = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = domp.parse(new ByteArrayInputStream(response.getBytes()));

        NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("code");
        for (int i = 0; i < errorNodes.getLength(); i++) {
            Element element = (Element) errorNodes.item(i);
            if(element.getTextContent() != null && !element.getTextContent().equals("")) {
                NodeList nodes = doc.getDocumentElement().getElementsByTagName("description");
                for (int j = 0; j < nodes.getLength(); j++) {
                    Element element2 = (Element) nodes.item(j);
                    throw new Exception(element2.getTextContent());
                }
            }
        }
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }
```

```
    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

9.8 Subscribing to change notifications – Deployment, OOB

The following simple sample program implemented in java demonstrates a REST client that registers for change notifications from CSM to receive Deployment and Out of Band (OOB) events. Please note that to receive change notifications, the client registers a specific **syslog service (IP and Port) to which CSM will sent 'asynchronous' notifications**. To see the actual notifications, please use any open source Syslog server or implement a simple UDP listener at the specified IP/Port (see SyslogServer element below) to list the notification content.

Use the client.properties file as listed below:

```
USER=admin
PASSWORD=admin
HOST=localhost
XML_REQUEST=<?xml version=\"1.0\" encoding=\"UTF-8\"?>\
<csm:eventSubRequest xmlns:csm=\"csm\">\
  <op>add</op>\
  <subscriptionId>123454</subscriptionId>\
  <eventFilterItem>\
  <filterEventType>syslog</filterEventType>\
  <filterEventFormat>xml</filterEventFormat>\
  <filterEventCategory>configChange</filterEventCategory>\
  </eventFilterItem>\
  <syslogServer>\
     <port>514</port>\
     <destAddress>10.10.10.10</destAddress>\
  </syslogServer>\
</csm:eventSubRequest>

# Set LOGIN_REQUIRED to true if the URI supplied
# requires login to be done as a prerequisite.
LOGIN_REQUIRED=true
URI=https://localhost/nbi/eventservice/eventSubscription
```

After compiling, use the following command to run the program:

```
Command Prompt> java RestClient <path_to_client.properties> [<uri>]
```

Class `RestClient.java`

```
/**
 * Sample program to subscribe to change events and get more details on latest change.
 */
import java.io.ByteArrayInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URI;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;
import java.util.Properties;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.CookieStore;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
```

```
import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;


public class RestClient {

    public static CookieStore ascookie = null;
    public static DefaultHttpClient httpclient;

    static{
        initSSL();
    }


    private static void initSSL() {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, new TrustManager[] { new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    System.out.println("getAcceptedIssuers =============");
                    return null;
                }
                public void checkClientTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkClientTrusted =============");
                }
                public void checkServerTrusted(X509Certificate[] certs, String authType) {
                    System.out.println("checkServerTrusted =============");
                }
            }}, new SecureRandom());


            SSLSocketFactory sf = new SSLSocketFactory(sslContext);

            Scheme httpsScheme = new Scheme("https", 443, sf);
            SchemeRegistry schemeRegistry = new SchemeRegistry();
            schemeRegistry.register(httpsScheme);
            HttpParams params = new BasicHttpParams();
            ClientConnectionManager cm = new ThreadSafeClientConnManager(params, schemeRegistry);
            httpclient = new DefaultHttpClient(cm, params);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    /**
     * This method will send the XML payload and return the XML response as a string.
     * @param uri
     * @param host
     * @param isCookieNeeded
     * @return
     * @throws IOException
     * @throws ClientProtocolException
     */
    public void doPost (URI uri, String payload, String host, boolean isCookieNeeded) throws
Exception {

        HttpResponse httpresponse = null;
        HttpPost httppost = new HttpPost (uri);

        if (isCookieNeeded) {
```

```
            httpclient.setCookieStore(ascookie);
        }
        httppost.addHeader("Content-Type", "text/xml");
        StringEntity strEntity = new StringEntity (payload, "UTF-8");
        httppost.setEntity(strEntity);
        System.out.println("Calling : "+uri.toString());
        httpresponse = httpclient.execute(httppost);
        ascookie = httpclient.getCookieStore();
        processResponse(httpresponse);

    }

    public void processResponse (HttpResponse httpresponse) throws Exception, IOException,
SAXException {
        HttpEntity ent = httpresponse.getEntity();
        String response = EntityUtils.toString(ent);
        DocumentBuilder domp = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = domp.parse(new ByteArrayInputStream(response.getBytes()));

        NodeList errorNodes = doc.getDocumentElement().getElementsByTagName("code");
        for (int i = 0; i < errorNodes.getLength(); i++) {
            Element element = (Element) errorNodes.item(i);
            if(element.getTextContent() != null && !element.getTextContent().equals("")) {
                NodeList nodes = doc.getDocumentElement().getElementsByTagName("description");
                for (int j = 0; j < nodes.getLength(); j++) {
                    Element element2 = (Element) nodes.item(j);
                    throw new Exception(element2.getTextContent());
                }
            }
        }
        if(response != null && response.trim().length() != 0){
            StatusLine sl;
            if ((sl = httpresponse.getStatusLine()) != null) {
                if (sl.getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
                    System.out.println("Hit Authorization exception");
                    System.out.println(response);
                    //Do something...
                }else if (sl.getStatusCode() == HttpStatus.SC_OK) {
                    System.out.println("The request is a success...");
                    System.out.println(response);
                    //Do something...
                }else{
                    System.out.println("Some issue, Obtained HTTP status code
:"+sl.getStatusCode());
                    System.out.println(response);
                    //Do something...
                }
            }
        }
    }

    /**
     * Main method processing the request/response
     * @param args
     */
    public static void main(String[] args){
        try{
            //Load the basic properties
            FileInputStream fis = null;
            Properties prop = new Properties();
            fis = new FileInputStream(args[0]);
            prop.load(fis);

            String host = prop.getProperty("HOST");
            String payload = prop.getProperty("XML_REQUEST");
            String username = prop.getProperty("USER");
            String password = prop.getProperty("PASSWORD");
            String path = prop.getProperty("URI");
            //If URI is not passed on commandline see if its defined in properties file
```

```
            URI uri = new URI( (args.length == 2)?args[1] : path);
            String temp = prop.getProperty("LOGIN_REQUIRED");
            boolean autoLogin = false;
            if(null != temp && temp.trim().length() != 0){
                autoLogin = Boolean.valueOf(temp);
            }

            RestClient client = new RestClient();
            if(uri.toString().endsWith("login")){
                client.doPost(uri, payload, host, false);
            }else{
                //Step 1 :
                if(autoLogin){
                    String login_payload = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><csm:loginRequest
xmlns:csm=\"csm\"><protVersion>1.0</protVersion><reqId>123</reqId><username>"+username+"</usernam
e><password>"+password+"</password></csm:loginRequest>";
                    client.doPost(new URI("https://"+host+"/nbi/login"), login_payload, host,
false);
                }
                //Step 2:
                client.doPost(uri, payload, host, true);
            }
        }catch(Exception ex){
            System.out.println(ex.getMessage()); usage();}
    }

    public static void usage(){
        System.out.println("Please check the data entered in the properties file");
        System.out.println("Usage : ");
        System.out.println("java RestClient <path_to_client.properties> [<uri>]");
    }
}
```

10 Troubleshooting (Common Scenarios)

Symptom: Certificate errors when communicating with the server

The CSM server uses self-signed certificates. Change the client program to accept self-signed certificates. Also API access is only available over HTTPS and HTTP access is disabled for security reasons.

Symptom: Authentication errors even after a successful login

Intermittent session errors will be seen if the session has timed out or has been invalidated. Default session inactivity timeout is 15 minutes. The session can be kept alive using a ping or heartbeat mechanism.

Symptom: Inline IP entered in the CSM GUI are retrieved as objects

CSM automatically encapsulates certain inline values (IP's, Interface names etc.) inside a policy object for convienience.

Symptom: Data visible in the CSM client UI are not seen in the response data when fetched via the API

This can be due to various reasons. Some of them are listed below:

- API only returns committed data (i.e. all changes must be submitted). So check if all data that is visible in the client is committed.
- In some cases CSM specific settings are not returned in the response as it may not be relevant for API access.
- Check if the user has sufficient Role Based Access Control (RBAC) privileges to access the data

Symptom: Client does not receive any event notifications from the server

Some of the reasons for this problem:

- The event subscription has not been done or has been registered with incorrect syslog server that receives the notification
- Session used to register the event subscription has timed out. All event notifications are tied to a user session. If the user session is invalidated then all event notifications for that user session will be stopped.

Symptom: Invalid API license error even after successful installation of API license file

To resolve this issue, clear the cache from the browser and restart CSM.

11 XML Schema

The XML schema is broken into four files.

11.1 Common XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="csm" targetNamespace="csm">
	<xs:simpleType name="ObjectIdentifier">
		<xs:restriction base="xs:string">
			<xs:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="ObjectIdentifierList">
		<xs:sequence>
			<xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="BaseObject">
		<xs:sequence>
			<xs:element name="gid" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
			<xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
			<xs:element name="lastUpdateTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
			<xs:element name="parentGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
			<xs:element name="updatedByUser" type="xs:string" minOccurs="0" maxOccurs="1"/>
			<xs:element name="lastCommitTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
			<xs:element name="ticketId" type="xs:string" minOccurs="0" maxOccurs="1"/>
			<xs:element name="activityName" type="xs:string" minOccurs="0" maxOccurs="1"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="BaseError">
		<xs:sequence>
			<xs:element name="code" type="xs:unsignedLong" maxOccurs="1"/>
			<xs:element name="description" type="xs:string" maxOccurs="1"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="BaseReqResp">
		<xs:sequence>
			<xs:element name="protVersion" type="xs:double" minOccurs="0" maxOccurs="1"/>
			<xs:element name="reqId" type="xs:string" minOccurs="0" maxOccurs="1"/>
			<xs:element name="startIndex" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
			<xs:element name="endIndex" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
			<xs:element name="totalCount" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
			<xs:element name="error" type="BaseError" minOccurs="0" maxOccurs="1"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="EntityDescriptor">
		<xs:sequence>
			<xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
			<xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
		</xs:sequence>
	</xs:complexType>
	<xs:element name="device" type="Device"/>
	<xs:complexType name="Device">
		<xs:complexContent>
			<xs:extension base="BaseObject">
```

```
                        <xs:sequence>
                            <xs:element name="osType" type="OSType" minOccurs="1" maxOccurs="1"/>
                            <xs:element name="osVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
                            <xs:element name="imageName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="sysObjectID" type="xs:string" minOccurs="1" maxOccurs="1"/>
                            <xs:element name="fullConfig" type="xs:string" minOccurs="0" maxOccurs="1"/>
                            <xs:element name="mgmtInterface" type="Interface" minOccurs="0" maxOccurs="1"/>
                            <xs:element name="interfaceList" type="InterfaceList" minOccurs="0" maxOccurs="1"/>
                            <xs:element name="virtualContextList" type="Device" minOccurs="0"
maxOccurs="unbounded"/>
                            <xs:element name="configState" type="ConfigurationState" minOccurs="0"
maxOccurs="1"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="PortIdentifier">
                <xs:sequence>
                    <!-- for non-modular chassis or chassis with a continuous port numbering scheme slot/module are
not included -->
                    <xs:element name="slotNum" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="moduleNum" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="portNum" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
            </xs:complexType>
            <xs:simpleType name="ProtocolPort">
                <xs:restriction base="xs:string"/>
            </xs:simpleType>
            <xs:complexType name="InterfaceList">
                <xs:sequence>
                    <xs:element name="interface" type="Interface" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="Interface">
                <xs:sequence>
                    <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="identifier" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="ipInterface" type="IPInterfaceAttrs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="macInterface" type="MACInterfaceAttrs" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="MACInterfaceAttrs">
                <xs:sequence>
                    <xs:element name="macAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="IPInterfaceAttrs">
                <xs:sequence>
                    <xs:element name="domainName" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="ipAddress" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="isNatAddress" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="realIpAddress" type="xs:string" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:complexType>
            <!--<xs:simpleType name="InterfaceType">
                <xs:restriction base="xs:token">
                    <xs:enumeration value="enet"/>
                    <xs:enumeration value="genet"/>
                    <xs:enumeration value="10genet"/>
```

```
                <xs:enumeration value="100genet"/>
                <xs:enumeration value="mgmt"/>
                <xs:enumeration value="tunnel"/>
                <xs:enumeration value="sensor"/>
                <xs:enumeration value="serial"/>
                <xs:enumeration value="vlan"/>
            </xs:restriction>
        </xs:simpleType> -->
        <xs:simpleType name="OSType">
            <xs:restriction base="xs:token">
                <xs:enumeration value="ios"/>
                <xs:enumeration value="fwsm"/>
                <xs:enumeration value="asa"/>
                <xs:enumeration value="ips"/>
                <xs:enumeration value="pix"/>
                <xs:enumeration value="undefined"/>

            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="IPTransportProtocol">
            <xs:restriction base="xs:token">
                <xs:enumeration value="TCP"/>
                <xs:enumeration value="UDP"/>
                <xs:enumeration value="IP"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ConfigurationState">
            <xs:restriction base="xs:token">
                <xs:enumeration value="undefined"/>
                <xs:enumeration value="committed"/>
                <xs:enumeration value="deployed"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DeviceCapability">
            <xs:restriction base="xs:token">
                <xs:enumeration value="firewall"/>
                <xs:enumeration value="ids"/>
                <xs:enumeration value="router"/>
                <xs:enumeration value="switch"/>
                <xs:enumeration value="*"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="DeviceGroup">
            <xs:complexContent>
                <xs:extension base="BaseObject">
                    <xs:sequence>
                        <xs:element name="path" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="device" type="Device" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="deviceGroup" type="DeviceGroup" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DeviceGroupPath">
            <xs:sequence>
                <xs:element name="pathItem" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:simpleType name="SubscriptionOperation">
```

```
            <xs:restriction base="xs:string">
                <xs:enumeration value="add"/>
                <xs:enumeration value="delete"/>
            </xs:restriction>
        </xs:simpleType>
        <!-- Common Service Methods
    -->
    <!-- Generic error -->
    <xs:element name="baseError" type="BaseError"/>
        <xs:element name="loginRequest" type="LoginRequest"/>
        <xs:complexType name="LoginRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence>
                        <xs:element name="username" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="password" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="heartbeatRequested" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
                        <xs:element name="callbackUrl" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="loginResponse" type="LoginResponse"/>
        <xs:complexType name="LoginResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence>
                        <xs:element name="serviceVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="sessionTimeoutInMins" type="xs:positiveInteger" minOccurs="1"
maxOccurs="1"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="heartbeatCallbackRequest" type="HeartbeatCallbackRequest"/>
        <xs:complexType name="HeartbeatCallbackRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="logoutRequest" type="LogoutRequest"/>
        <xs:complexType name="LogoutRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="logoutResponse" type="LogoutResponse"/>
        <xs:complexType name="LogoutResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="pingRequest" type="PingRequest"/>
        <xs:complexType name="PingRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="pingResponse" type="PingResponse"/>
```

```
    <xs:complexType name="PingResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="getServiceInfoRequest" type="GetServiceInfoRequest"/>
    <xs:complexType name="GetServiceInfoRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="getServiceInfoResponse" type="GetServiceInfoResponse"/>
    <xs:complexType name="GetServiceInfoResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence>
                    <xs:element name="serviceVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="serviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="serviceDesc" type="xs:string" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

11.2 Config XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2011 (x64) (http://www.altova.com) by Rakesh (Cisco) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="csm" targetNamespace="csm">
   <xs:include schemaLocation="common.xsd"/>
   <xs:complexType name="BasePolicy" >
      <xs:complexContent>
         <xs:extension base="BaseObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
               <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="orderId" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
               <xs:element name="isMandatoryAggregation" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="eventCorrelationID" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="configState" type="ConfigurationState" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="BasePolicyObject">
      <xs:complexContent>
         <xs:extension base="BaseObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
               <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="nodeGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
               <xs:element name="isProperty" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
               <xs:element name="subType" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="isGroup" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
               <xs:element name="refGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1"/>
               <xs:element name="eventCorrelationID" type="xs:string" minOccurs="0" maxOccurs="1"/>
               <xs:element name="configState" type="ConfigurationState" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="NetworkInterfaceObjectsRefs">
      <xs:sequence>
         <xs:element name="networkObjectGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1"/>
         <xs:element name="interfaceRoleObjectGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1"/>
         <xs:choice>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="ipData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="NetworkObjectsRefs">
      <xs:sequence>
         <xs:element name="networkObjectGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1"/>
         <xs:choice>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="ipData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="SecurityGrpObjectsRef">
      <xs:sequence>
```

```
        <xs:element name="securityGrpObjectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
        <xs:element name="secName" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="secTag" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="SecurityGrpObjectsRefs">
    <xs:sequence>
        <xs:element name="securityTag" type="SecurityGrpObjectsRef" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="IdentityUserGrpObjectsRefs">
    <xs:sequence>
        <xs:element name="identityUserGrpObjectGIDs" type="ObjectIdentifierList" minOccurs="0" maxOccurs="1"/>
        <xs:element name="userNameData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="userGroupData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="NetworkObjectRefs">
    <xs:sequence>
        <xs:element name="networkObjectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
        <xs:choice>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="ipData" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:choice>
<xs:element name="interfaceKeyword" type="xs:string" fixed="interface" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="NetworkOrIPRef">
    <xs:choice>
        <xs:element name="hostOrNetworkObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
        <xs:choice>
            <xs:element name="ipv4Data" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="ipData" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:choice>
    </xs:choice>
</xs:complexType>
<xs:complexType name="NetworkPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence>
                <xs:choice>
                    <xs:element name="ipv4Data" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="ipData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                </xs:choice>
                <xs:element name="fqdnData" minOccurs="0" maxOccurs="1">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="value" type="xs:string" minOccurs="0"
                                                maxOccurs="1" />
                                            <xs:element name="fqdnType" minOccurs="0" maxOccurs="1">
                                                <xs:simpleType>
                                                    <xs:restriction base="xs:token">
                                                        <xs:enumeration value="IPv4 Only" />
                                                        <xs:enumeration value="IPv6 Only" />
                                                        <xs:enumeration value="Default" />
                                                    </xs:restriction>
                                                </xs:simpleType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
```

```
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
    </xs:complexType>
<xs:complexType name="IdentityUserGroupPolicyObject">
    <xs:complexContent>
        <xs:extension base="BasePolicyObject">
            <xs:sequence>
                <xs:element name="userNameData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="userGroupData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
    <xs:complexType name="SecurityGroupPolicyObject">
            <xs:complexContent>
                <xs:extension base="BasePolicyObject">
                    <xs:sequence>
                        <xs:element name="securityTag" type="SecurityGrpObjectsRef"
                            minOccurs="0" maxOccurs="unbounded" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
    </xs:complexType>

    <xs:simpleType name="OperatorType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="gt" />
                <xs:enumeration value="lt" />
                <xs:enumeration value="eq" />
                <xs:enumeration value="neq" />
            </xs:restriction>
    </xs:simpleType>



    <xs:complexType name="PortListPolicyObject">
            <xs:complexContent>
                <xs:extension base="BasePolicyObject">
                    <xs:sequence minOccurs="0" maxOccurs="1">
                    <xs:element name="port" type="PortRange" minOccurs="0" maxOccurs="unbounded" />

                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PortRange">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="operator" type="OperatorType" minOccurs="0" maxOccurs="1" />
                <xs:element name="startPort" type="PortIdentifier" minOccurs="1" maxOccurs="1"/>
                <xs:element name="endPort" type="PortIdentifier" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ServiceParameters">
    <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="protocol" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="sourcePort" minOccurs="0" maxOccurs="1">
```

```
            <xs:complexType>
               <xs:choice>
                  <xs:element name="port" type="ProtocolPort"/>
                  <xs:element name="portRefGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               </xs:choice>
            </xs:complexType>
         </xs:element>
         <xs:element name="destinationPort" minOccurs="0" maxOccurs="1">
            <xs:complexType>
               <xs:choice>
                  <xs:element name="port" type="ProtocolPort"/>
                  <xs:element name="portRefGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               </xs:choice>
            </xs:complexType>
         </xs:element>
         <xs:element name="icmpMessage" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ServicePolicyObject">
      <xs:complexContent>
         <xs:extension base="BasePolicyObject">
            <xs:sequence minOccurs="0" maxOccurs="1">
               <xs:element name="serviceParameters" type="ServiceParameters" minOccurs="1"
maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:extension>

      </xs:complexContent>

   </xs:complexType>
   <xs:complexType name="InterfaceRolePolicyObject">
      <xs:complexContent>
         <xs:extension base="BasePolicyObject">
            <xs:sequence>
               <xs:element name="pattern" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="TimeRangePolicyObject">
      <xs:complexContent>
         <xs:extension base="BasePolicyObject">
            <xs:sequence minOccurs="1" maxOccurs="1">
               <xs:element name="startTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
               <xs:element name="endTime" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
               <xs:element name="recurrence" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                     <xs:choice>
                        <xs:element name="dayOfWeekInterval">
                           <xs:complexType>
                              <xs:sequence minOccurs="1" maxOccurs="1">
                                 <xs:element name="dayOfWeek" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                 <xs:element name="startTime" type="xs:time" minOccurs="1" maxOccurs="1"/>
                                 <xs:element name="endTime" type="xs:time" minOccurs="1" maxOccurs="1"/>
                              </xs:sequence>
                           </xs:complexType>
                        </xs:element>
                        <xs:element name="weeklyInterval">
                           <xs:complexType>
                              <xs:sequence minOccurs="1" maxOccurs="1">
```

```
                                <xs:element name="startDay" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="startTime" type="xs:time" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="endDay" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="endTime" type="xs:time" minOccurs="1" maxOccurs="1"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="SLAMonitorPolicyObject">
<xs:complexContent>
<xs:extension base="BasePolicyObject">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="slaId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
<xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
<xs:element name="monitoredAddress" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="dataSizeInBytes" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
<xs:element name="thresholdInMilliSeconds" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
<xs:element name="timeoutInMilliSeconds" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
<xs:element name="frequencyInSeconds" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
<xs:element name="toS" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
<xs:element name="numberOfPackets" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="StandardACEPolicyObject">
<xs:complexContent>
<xs:extension base="BasePolicyObject">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="networkGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
<xs:element name="doLogging" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
<xs:element name="permit" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="ExtendedACEPolicyObject">
<xs:complexContent>
<xs:extension base="BasePolicyObject">
<xs:sequence>
<xs:element name="sourceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
<xs:element name="destinationGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
<xs:element name="serviceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
<xs:element name="doLogging" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="logInterval" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="logLevel" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="logOption" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="permit" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="ACLPolicyObject">
<xs:complexContent>
```

```
        <xs:extension base="BasePolicyObject">
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="references" minOccurs="1" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
                  <xs:element name="sequenceNumber" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                  <xs:choice>
                    <xs:element name="aclObjectReferenceGID" type="ObjectIdentifier"/>
                    <xs:element name="aceReferenceGID" type="ObjectIdentifier"/>
                  </xs:choice>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="DeviceAccessRuleFirewallPolicy">
      <xs:complexContent>
        <xs:extension base="BasePolicy">
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="isEnabled" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="direction" minOccurs="0" maxOccurs="1">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="in"/>
                  <xs:enumeration value="out"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="permit" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sectionName" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="policyName" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="interfaceRoleObjectGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="users" type="IdentityUserGrpObjectsRefs" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sources" type="NetworkInterfaceObjectsRefs" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinations" type="NetworkInterfaceObjectsRefs" minOccurs="0" maxOccurs="1"/>
            <xs:element name="services">
              <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
                  <xs:element name="serviceObjectGIDs" type="ObjectIdentifierList" minOccurs="0"
maxOccurs="1"/>
                  <xs:element name="serviceParameters" type="ServiceParameters" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="logOptions" minOccurs="0" maxOccurs="1">
              <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
                  <xs:element name="isFirewallLoggingEnabled" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
                  <xs:choice>
                    <xs:element name="isDefaultLogging" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                    <xs:sequence minOccurs="1" maxOccurs="1">
                      <xs:element name="loggingInterval" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
                      <xs:element name="loggingLevel" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                  </xs:choice>
```

```
                        <xs:element name="isIOSLoggingEnabled" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="isLogInput" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="iosOptions" minOccurs="0" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:enumeration value="None"/>
                        <xs:enumeration value="Established"/>
                        <xs:enumeration value="Fragment"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="timeRangeObjectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DeviceAccessRuleUnifiedFirewallPolicy">
      <xs:complexContent>
         <xs:extension base="DeviceAccessRuleFirewallPolicy">
            <xs:sequence minOccurs="1" maxOccurs="1">
               <xs:element name="sourceSG" type="SecurityGrpObjectsRefs" minOccurs="0" maxOccurs="1"/>
               <xs:element name="destinationSG" type="SecurityGrpObjectsRefs" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DeviceStaticRoutingFirewallPolicy">
      <xs:complexContent>
         <xs:extension base="BasePolicy">
            <xs:sequence>
               <xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               <xs:element name="networks" type="NetworkObjectsRefs" minOccurs="1" maxOccurs="1"/>
               <xs:element name="gateway" type="NetworkObjectRefs" minOccurs="0" maxOccurs="1"/>
               <xs:element name="metric" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
               <xs:element name="tunnelled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="slaMonitorGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DeviceStaticRoutingRouterPolicy">
      <xs:complexContent>
         <xs:extension base="BasePolicy">
            <xs:sequence>
               <xs:element name="destinationNetwork" minOccurs="1" maxOccurs="1">
                  <xs:complexType>
                     <xs:choice>
                        <xs:element name="useAsDefaultRoute" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="prefix" type="NetworkOrIPRef" minOccurs="1" maxOccurs="1"/>
                     </xs:choice>
                  </xs:complexType>
               </xs:element>
               <xs:element name="fowarding" minOccurs="1" maxOccurs="1">
                  <xs:complexType>
                     <xs:choice>
                        <xs:element name="forwardingInterfaceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
```

```
                        <xs:element name="forwardingIPAddress" type="NetworkOrIPRef" minOccurs="1"
maxOccurs="1"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="distanceMetric" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isPermanentRoute" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="DeviceBGPRouterPolicy">
  <xs:complexContent>
   <xs:extension base="BasePolicy">
    <xs:sequence>
     <xs:element name="asNumber" type="xs:unsignedLong" minOccurs="1" maxOccurs="1"/>
     <xs:element name="networks" type="NetworkObjectsRefs" minOccurs="0" maxOccurs="1"/>
     <xs:element name="neighbors" minOccurs="1" maxOccurs="1">
      <xs:complexType>
       <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element name="ipAddress" type="NetworkObjectsRefs" minOccurs="1" maxOccurs="1"/>
        <xs:element name="asNumber" type="xs:unsignedLong" minOccurs="1" maxOccurs="1"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="autoSummary" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
     <xs:element name="synchronization" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
     <xs:element name="logNeighbor" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
     <xs:element name="redistributionEntry" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="protocol" minOccurs="1" maxOccurs="1">
         <xs:complexType>
          <xs:choice>
           <xs:element name="static" minOccurs="1" maxOccurs="1">
                        <xs:simpleType>
                         <xs:restriction base="xs:string">
                          <xs:enumeration value="IP"/>
                          <xs:enumeration value="OSI"/>
                         </xs:restriction>
                        </xs:simpleType>
           </xs:element>
           <xs:element name="connected" type="xs:string" minOccurs="1" maxOccurs="1"/>
           <xs:element name="rip" type="xs:string" minOccurs="1" maxOccurs="1"/>
           <xs:element name="eigrp" minOccurs="1" maxOccurs="1">
            <xs:complexType>
             <xs:sequence>
              <xs:element name="asNumber" type="xs:unsignedInt" minOccurs="1"
maxOccurs="1"/>
             </xs:sequence>
            </xs:complexType>
           </xs:element>
           <xs:element name="ospf" minOccurs="1" maxOccurs="1">
            <xs:complexType>
             <xs:sequence>
              <xs:element name="processId" type="xs:unsignedInt" minOccurs="1"
maxOccurs="1"/>
              <xs:element name="match" minOccurs="0" maxOccurs="unbounded">
                                <xs:simpleType>
                                 <xs:restriction base="xs:string">
```

```
                                                    <xs:enumeration value="Internal"/>
                                                    <xs:enumeration value="External1"/>
                                                    <xs:enumeration value="External2"/>
                                                    <xs:enumeration value="NSSAExternal1"/>
                                                    <xs:enumeration value="NSSAExternal2"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
                <xs:element name="metric" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="InterfaceNATRouterPolicy">
<xs:complexContent>
<xs:extension base="BasePolicy">
<xs:sequence>
<xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
<xs:element name="isNatInside" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="InterfaceNATStaticRulesRouterPolicy">
<xs:complexContent>
<xs:extension base="BasePolicy">
<xs:sequence>
<xs:element name="staticRuleType" minOccurs="1" maxOccurs="1">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Static Host"/>
<xs:enumeration value="Static Network"/>
<xs:enumeration value="Static Port"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="original" type="NetworkOrIPRef" minOccurs="1" maxOccurs="1"/>
<xs:element name="translated" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:choice>
<xs:element name="originalIP" type="NetworkOrIPRef" minOccurs="1" maxOccurs="1"/>
<xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="portRedirection" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="protocol" type="xs:string" minOccurs="1" maxOccurs="1"/>
<xs:element name="localPort" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
<xs:element name="globalPort" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
```

```
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="settings" minOccurs="0" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="noAlias" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="noPayload" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="createExtTransEntry" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="InterfaceNATDynamicRulesRouterPolicy">
      <xs:complexContent>
        <xs:extension base="BasePolicy">
          <xs:sequence>
            <xs:element name="trafficFlowAclObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
            <xs:element name="translated" minOccurs="1" maxOccurs="1">
              <xs:complexType>
                <xs:choice>
                  <xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="addressPool" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
                </xs:choice>
              </xs:complexType>
            </xs:element>
            <xs:element name="settings" minOccurs="0" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="enablePortTrans" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="noTransVPN" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="DeviceNATTimeoutsRouterPolicy">
      <xs:complexContent>
        <xs:extension base="BasePolicy">
          <xs:sequence>
            <xs:element name="maxEntriesInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="timeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="udpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="dnsTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="tcpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="finRstTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="icmpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="pptpTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xs:element name="synTimeoutInSecs" type="xs:unsignedLong" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!--
      Firewall NAT
```

```
    -->
    <!--  Reusable Firewall advanced options -->
    <xs:complexType name="FirewallNATAdvancedOptions">
        <xs:sequence>
            <xs:element name="isTransDNSReplies" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="maxTCPConnPerRule" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
            <xs:element name="maxUDPConnPerRule" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
            <xs:element name="maxEmbConnections" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
            <xs:element name="randomizeSeqNum" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <!-- Reusable NAT Type -->
    <xs:simpleType name="NATType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Static"/>
            <xs:enumeration value="Dynamic"/>
        </xs:restriction>
    </xs:simpleType>
    <!-- Reusable Protocol Type -->
    <xs:complexType name="InterfaceNATAddressPoolFirewallPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="interfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="poolId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="ipAddressRange" type="NetworkObjectsRefs" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="interfaceKeyword" type="xs:string" fixed="interface" minOccurs="0"
maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="DeviceNATTransOptionsFirewallPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="isEnableTrafficWithoutTrans" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                    <!--<xs:element name="isEnableVPNTrafficWithoutTrans" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>-->
                    <xs:element name="isXlateByPass" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="InterfaceNATTransExemptionsFirewallPolicy">
        <xs:complexContent>
            <xs:extension base="BasePolicy">
                <xs:sequence>
                    <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="isExempt" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="original" type="NetworkInterfaceObjectsRefs" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="outsideNAT" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="destinations" type="NetworkInterfaceObjectsRefs" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="fwsmAdvancedOptions" type="FirewallNATAdvancedOptions" minOccurs="0"
maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

```
   <xs:complexType name="InterfaceNATDynamicRulesFirewallPolicy">
      <xs:complexContent>
         <xs:extension base="BasePolicy">
            <xs:sequence>
               <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               <xs:element name="poolId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
               <xs:element name="original" type="NetworkObjectsRefs" minOccurs="1" maxOccurs="1"/>
               <xs:element name="outsideNAT" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="advancedOptions" type="FirewallNATAdvancedOptions" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="InterfaceNATPolicyDynamicRulesFirewallPolicy">
      <xs:complexContent>
         <xs:extension base="BasePolicy">
            <xs:sequence>
               <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               <xs:element name="poolId" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
               <xs:element name="original" type="NetworkInterfaceObjectsRefs" minOccurs="1" maxOccurs="1"/>
               <xs:element name="outsideNAT" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="destinations" type="NetworkInterfaceObjectsRefs" minOccurs="1" maxOccurs="1"/>
               <xs:element name="services" minOccurs="1" maxOccurs="1">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="serviceData" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="serviceObjectGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="unbounded"/>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="advancedOptions" type="FirewallNATAdvancedOptions" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="InterfaceNATStaticRulesFirewallPolicy">
      <xs:complexContent>
         <xs:extension base="BasePolicy">
            <xs:sequence>
               <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="translationType" minOccurs="1" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:enumeration value="NAT"/>
                        <xs:enumeration value="PAT"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               <xs:element name="mappedInterfaceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
               <xs:element name="original" type="NetworkOrIPRef" minOccurs="1" maxOccurs="1"/>
               <xs:element name="translated" type="NetworkObjectRefs" minOccurs="1" maxOccurs="1"/>
               <xs:element name="policyNAT" minOccurs="0" maxOccurs="1">
                  <xs:complexType>
                     <xs:sequence>
```

```
                        <xs:element name="destAddress" type="NetworkObjectsRefs" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="services" minOccurs="1" maxOccurs="1">
                           <xs:complexType>
                              <xs:sequence>
                                 <xs:element name="serviceData" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                                 <xs:element name="serviceObjectGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="unbounded"/>
                              </xs:sequence>
                           </xs:complexType>
                        </xs:element>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="protocol" type="IPTransportProtocol" minOccurs="1" maxOccurs="1"/>
               <xs:element name="originalPort" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
               <xs:element name="translatedPort" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
               <xs:element name="advancedOptions" type="FirewallNATAdvancedOptions" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="PatOptions">
      <xs:sequence>
            <xs:element name="patAddressPool" minOccurs="0" maxOccurs="1">
                  <xs:complexType>
                        <xs:choice>
                              <xs:element name="patPoolAddressGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1"/>
                              <xs:element name="interfaceKeyword" type="xs:string" fixed="interface" minOccurs="0"
maxOccurs="1"/>
                        </xs:choice>
                  </xs:complexType>
            </xs:element>
            <xs:element name="isPatAllocatedInRoundRobin" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="InterfaceNATManualFirewallPolicy">
      <xs:complexContent>
         <xs:extension base="BasePolicy">
            <xs:sequence>
               <xs:element name="isRuleEnabled" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
               <xs:element name="section" minOccurs="1" maxOccurs="1">
                  <xs:simpleType>
                     <xs:restriction base="xs:unsignedInt">
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                        <xs:enumeration value="3"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="realInterface" minOccurs="0" maxOccurs="1">
                  <xs:complexType>
                     <xs:choice>
                        <xs:element name="realInterfaceGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="realInterfaceName" type="xs:string" minOccurs="0" maxOccurs="1"/>
                     </xs:choice>
                  </xs:complexType>
               </xs:element>
```

```
            <xs:element name="mappedInterface" minOccurs="0" maxOccurs="1">
               <xs:complexType>
                  <xs:choice>
                     <xs:element name="mappedInterfaceGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1"/>
                     <xs:element name="mappedInterfaceName" type="xs:string" minOccurs="0" maxOccurs="1"/>
                  </xs:choice>
               </xs:complexType>
            </xs:element>
            <xs:element name="source" minOccurs="1" maxOccurs="1">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="natType" type="NATType" minOccurs="1" maxOccurs="1"/>
                     <xs:element name="originalObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                                          <xs:element name="translated" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="objectGID" type="NetworkObjectRefs" minOccurs="0"
maxOccurs="1"/>
                              <xs:element name="patPool" type="PatOptions" minOccurs="0" maxOccurs="1"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="destination" minOccurs="0" maxOccurs="1">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="natType" type="NATType" fixed="Static" minOccurs="1" maxOccurs="1"/>
                                          <xs:element name="originalObject" type="NetworkObjectRefs"
minOccurs="0" maxOccurs="1"/>
                                          <xs:element name="translatedObjectGID" type="ObjectIdentifier"
minOccurs="1" maxOccurs="1"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="service" minOccurs="0" maxOccurs="1">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="originalObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                     <xs:element name="transObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="isTransDNSReplies" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="direction" minOccurs="0" maxOccurs="1">
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="Unidirectional"/>
                     <xs:enumeration value="Bidirectional"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element name="isNoProxyARP" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isRouteLookUp" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

```
  <xs:complexType name="InterfaceNATObjectFirewallPolicy">
    <xs:complexContent>
      <xs:extension base="BasePolicy">
        <xs:sequence>
          <xs:element name="section" fixed="2" minOccurs="1" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="xs:unsignedInt">
                <xs:enumeration value="1"/>
                <xs:enumeration value="2"/>
                <xs:enumeration value="3"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="realInterface" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="mappedInterface" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="natType" type="NATType" minOccurs="1" maxOccurs="1"/>
          <xs:element name="originalObjectGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
          <xs:element name="translated" minOccurs="0" maxOccurs="1">
                  <xs:complexType>
                      <xs:sequence>
                          <xs:element name="objectGID" type="NetworkObjectRefs" minOccurs="0"
maxOccurs="1"/>
                          <xs:element name="patPool" type="PatOptions" minOccurs="0" maxOccurs="1"/>
                      </xs:sequence>
                  </xs:complexType>
              </xs:element>

          <xs:element name="isTransDNSReplies" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
          <xs:element name="isNoProxyARP" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
          <xs:element name="isRouteLookUp" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
          <xs:element name="service" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="protocol" type="IPTransportProtocol" minOccurs="1" maxOccurs="1"/>
                <xs:element name="originalPort" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
                <xs:element name="transPort" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- Config Service Methods
  -->
  <xs:element name="groupListRequest" type="GroupListRequest"/>
  <xs:complexType name="GroupListRequest">
    <xs:complexContent>
      <xs:extension base="BaseReqResp">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="includeEmptyGroups" type="xs:boolean"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="InterfaceNAT64ManualFirewallPolicy">
    <xs:complexContent>
      <xs:extension base="InterfaceNATManualFirewallPolicy">
        <xs:sequence minOccurs="1" maxOccurs="1">
```

```
            <xs:element name="isInterfaceIpv6" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isNetToNet" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:element name="groupListResponse" type="GroupListResponse"/>
    <xs:complexType name="GroupListResponse">
      <xs:complexContent>
        <xs:extension base="BaseReqResp">
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="deviceGroup" type="DeviceGroup"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="deviceListByCapabilityRequest" type="DeviceListByCapabilityRequest"/>
    <xs:complexType name="DeviceListByCapabilityRequest">
      <xs:complexContent>
        <xs:extension base="BaseReqResp">
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="deviceCapability" type="DeviceCapability" minOccurs="1"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="deviceListResponse" type="DeviceListResponse"/>
    <xs:complexType name="DeviceListResponse">
      <xs:complexContent>
        <xs:extension base="BaseReqResp">
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="deviceId" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
                  <xs:element name="deviceCapability" type="DeviceCapability" minOccurs="1" maxOccurs="1"/>
                  <!-- ipv4address is made optional as there could be virtual contexts configured without ip -->
                  <xs:element name="deviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="ipv4Address" type="xs:string" minOccurs="0" maxOccurs="1"/>
                  <xs:element name="sysObjectID" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="InterfaceNAT64ObjectFirewallPolicy">
      <xs:complexContent>
        <xs:extension base="InterfaceNATObjectFirewallPolicy">
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="isInterfaceIpv6" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isNetToNet" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
```

```
<xs:element name="deviceListByGroupRequest" type="DeviceListByGroupRequest"/>
<xs:complexType name="DeviceListByGroupRequest">
   <xs:complexContent>
      <xs:extension base="BaseReqResp">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="deviceGroupPath" type="DeviceGroupPath"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="deviceConfigByGIDRequest" type="DeviceConfigByGIDRequest"/>
<xs:complexType name="DeviceConfigByGIDRequest">
   <xs:complexContent>
      <xs:extension base="BaseReqResp">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="deviceConfigByNameRequest" type="DeviceConfigByNameRequest"/>
<xs:complexType name="DeviceConfigByNameRequest">
   <xs:complexContent>
      <xs:extension base="BaseReqResp">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="deviceConfigResponse" type="DeviceConfigResponse"/>
<xs:complexType name="DeviceConfigResponse">
   <xs:complexContent>
      <xs:extension base="BaseReqResp">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="device" type="Device" minOccurs="1" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="policyConfigByNameRequest" type="PolicyConfigByNameRequest"/>
<xs:complexType name="PolicyConfigByNameRequest">
   <xs:complexContent>
      <xs:extension base="BaseReqResp">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="policyConfigResponse" type="PolicyConfigResponse"/>
<xs:complexType name="PolicyConfigResponse">
   <xs:complexContent>
      <xs:extension base="BaseReqResp">
         <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="statusCode" type="OperationStatus" minOccurs="1" maxOccurs="1" />
            <xs:element name="gid" type="ObjectIdentifier" minOccurs="0" maxOccurs="500" />
            <xs:element name="policy" type="BasePolicy" minOccurs="0" maxOccurs="500" />
```

```
            <xs:element name="errorInfo" type="BaseError" minOccurs="0" maxOccurs="500" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
     </xs:complexType>
  <xs:element name="policyListByDeviceGIDRequest" type="PolicyListByDeviceGIDRequest"/>
  <xs:complexType name="PolicyListByDeviceGIDRequest">
    <xs:complexContent>
      <xs:extension base="BaseReqResp">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="policyListDeviceResponse" type="PolicyListDeviceResponse"/>
  <xs:complexType name="PolicyListDeviceResponse">
    <xs:complexContent>
      <xs:extension base="BaseReqResp">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="policyList" minOccurs="1" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="policyDesc" type="EntityDescriptor" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="policyNamesByTypeRequest" type="PolicyNamesByTypeRequest"/>
  <xs:complexType name="PolicyNamesByTypeRequest">
    <xs:complexContent>
      <xs:extension base="BaseReqResp">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="policyConfigByDeviceGIDRequest" type="PolicyConfigByDeviceGIDRequest"/>
  <xs:complexType name="PolicyConfigByDeviceGIDRequest">
    <xs:complexContent>
      <xs:extension base="BaseReqResp">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
          <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
     <!-- for add / modify of any policy, currently allows only acls -->
     <xs:element name="setPolicyConfigRequest" type="SetPolicyConfigRequest" />
     <xs:complexType name="SetPolicyConfigRequest">
          <xs:complexContent>
               <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
```

```
                        <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                        <xs:choice minOccurs="1" maxOccurs="1">
                            <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
                        </xs:choice>
                        <xs:choice minOccurs="1" maxOccurs="1">
                            <xs:element name="deviceAccessRuleUnifiedFirewallPolicy"
type="DeviceAccessRuleUnifiedFirewallPolicy" minOccurs="0"
                                maxOccurs="500" />
                            <xs:element name="deviceAccessRuleFirewallPolicy"
type="DeviceAccessRuleFirewallPolicy" minOccurs="0" maxOccurs="500" />
                            <xs:element name="firewallACLSettingsPolicy" type="FirewallACLSettingsPolicy"
minOccurs="0" maxOccurs="500">
                            </xs:element>
                        </xs:choice>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="deletePolicyConfigRequest" type="DeletePolicyConfigRequest" />
        <xs:complexType name="DeletePolicyConfigRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                        <xs:choice minOccurs="1" maxOccurs="1">
                            <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
                        </xs:choice>
                        <xs:element name="policyGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="500"
/>
                        <xs:element name="policyType" type="PolicyConfigType" minOccurs="1" maxOccurs="1"
/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    <xs:element name="policyNamesResponse" type="PolicyNamesResponse"/>
    <xs:complexType name="PolicyNamesResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="policyType" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="policy" minOccurs="1" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="policyName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                <xs:element name="deviceAssignments" minOccurs="0" maxOccurs="1">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="device" minOccurs="0" maxOccurs="unbounded">
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1"/>
```

```
                                    <xs:element name="deviceName" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="policyConfigDeviceResponse" type="PolicyConfigDeviceResponse"/>
 <xs:complexType name="PolicyConfigDeviceResponse">
  <xs:complexContent>
   <xs:extension base="BaseReqResp">
    <xs:sequence minOccurs="1" maxOccurs="1">
     <xs:element name="policy" minOccurs="1" maxOccurs="1">
      <xs:complexType>
       <xs:sequence minOccurs="1" maxOccurs="1">
        <!-- ALL DEFINED POLICY TYPES -->
        <xs:element name="deviceAccessRuleFirewallPolicy" type="DeviceAccessRuleFirewallPolicy"
minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="deviceAccessRuleUnifiedFirewallPolicy"
type="DeviceAccessRuleUnifiedFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="firewallACLSettingsPolicy" type="FirewallACLSettingsPolicy" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="deviceStaticRoutingRouterPolicy" type="DeviceStaticRoutingRouterPolicy"
minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="deviceStaticRoutingFirewallPolicy" type="DeviceStaticRoutingFirewallPolicy"
minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="deviceBgpRouterPolicy" type="DeviceBGPRouterPolicy" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="interfaceNATRouterPolicy" type="InterfaceNATRouterPolicy" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="interfaceNATStaticRulesRouterPolicy"
type="InterfaceNATStaticRulesRouterPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interfaceNATDynamicRulesRouterPolicy"
type="InterfaceNATDynamicRulesRouterPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="deviceNATTimeoutsRouterPolicy" type="DeviceNATTimeoutsRouterPolicy"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="interfaceNATAddressPoolFirewallPolicy"
type="InterfaceNATAddressPoolFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="deviceNATTransOptionsFirewallPolicy"
type="DeviceNATTransOptionsFirewallPolicy" minOccurs="0" maxOccurs="1"/>
        <xs:element name="interfaceNATTransExemptionsFirewallPolicy"
type="InterfaceNATTransExemptionsFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interfaceNATDynamicRulesFirewallPolicy"
type="InterfaceNATDynamicRulesFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interfaceNATPolicyDynamicRulesFirewallPolicy"
type="InterfaceNATPolicyDynamicRulesFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interfaceNATStaticRulesFirewallPolicy"
type="InterfaceNATStaticRulesFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interfaceNATManualFirewallPolicy"
type="InterfaceNATManualFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interfaceNATObjectFirewallPolicy" type="InterfaceNATObjectFirewallPolicy"
minOccurs="0" maxOccurs="unbounded"/>
```

```
                        <xs:element name="interfaceNAT64ManualFirewallPolicy"
type="InterfaceNAT64ManualFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="interfaceNAT64ObjectFirewallPolicy"
type="InterfaceNAT64ObjectFirewallPolicy" minOccurs="0" maxOccurs="unbounded"/>

                        <!-- ..... all other policies .. -->
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="policyObject" minOccurs="1" maxOccurs="1">
                  <xs:complexType>
                     <xs:sequence minOccurs="1" maxOccurs="1">
                        <!-- ALL DEFINED POLICY OBJECT TYPES -->
                        <xs:element name="networkPolicyObject" type="NetworkPolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="portListPolicyObject" type="PortListPolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="servicePolicyObject" type="ServicePolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="interfaceRolePolicyObject" type="InterfaceRolePolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="timeRangePolicyObject" type="TimeRangePolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="slaMonitorPolicyObject" type="SLAMonitorPolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="aclPolicyObject" type="ACLPolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="stdAcePolicyObject" type="StandardACEPolicyObject" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="extendedACEPolicyObject" type="ExtendedACEPolicyObject"
minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="identityUserGroupPolicyObject" type="IdentityUserGroupPolicyObject"
minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="securityGroupPolicyObject" type="SecurityGroupPolicyObject"
minOccurs="0" maxOccurs="unbounded"/>
                        <!-- ..... all other policy objects .. -->
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="newCSMSessionRequest" type="CSMSessionRequest" />
      <xs:complexType name="CSMSessionRequest">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
                        <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:element name="csmSessionName" type="xs:string" minOccurs="0"
                                    maxOccurs="1" />
                              <xs:element name="csmSessionDescription" type="xs:string"
                                    minOccurs="0" maxOccurs="1" />
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="csmSessionResponse" type="CSMSessionResponse" />
      <xs:complexType name="CSMSessionResponse">
            <xs:complexContent>
                  <xs:extension base="BaseReqResp">
```

```
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                        <xs:element name="description" type="xs:string" minOccurs="0"
                        maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DeployDeviceAttrs">
            <xs:sequence>
            <xs:element name="DeployOptions" minOccurs="0"
                        maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="device" />
                        <xs:enumeration value="file" />
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="filePath" type="xs:string"
                        minOccurs="0" maxOccurs="1" />
                            <xs:element name="OOBdetectionbehavior" minOccurs="0"
                                maxOccurs="1">
                                <xs:simpleType>
                                    <xs:restriction base="xs:token">
                                        <xs:enumeration value="over write changes and show warnings" />
                                        <xs:enumeration value="cancel deployment" />
                                        <xs:enumeration value="don not check for changes" />
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                        </xs:sequence>

                    </xs:complexType>

        <xs:element name="deployConfigRequest" type="DeployConfigRequest" />
        <xs:complexType name="DeployConfigRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="deviceGID" type="ObjectIdentifierList"
                            minOccurs="1" maxOccurs="1" />
                        <xs:element name="deploymentOptions" type="DeployDeviceAttrs"
                            minOccurs="0" maxOccurs="1" />
                        <xs:element name="deploymentComments" type="xs:string"
                            minOccurs="0" maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="deploymentMessageOptions" type="ValidationDetails" />
        <xs:element name="csmSessionValidationResponse" type="CSMSessionValidationResponse" />
          <xs:complexType name="CSMSessionValidationResponse">
           <xs:complexContent>
                <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="statusCode" type="OperationStatus" minOccurs="1" maxOccurs="1" />
                <xs:element name=""approvalRequired" type="xs:boolean"minOccurs="1" maxOccurs="1" />
                <xs:element name="validationMessage" type="xs:string" minOccurs="1" maxOccurs="1" />
```

```
            <xs:element name="validationResults" type="ValidationResults" minOccurs="0" maxOccurs="unbounded"
/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
    </xs:complexType>

   <xs:complexType name="ValidationResults">
        <xs:sequence minOccurs="1" maxOccurs="unbounded">
          <xs:element name="validationResult" type="ValidationResult" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ValidationResult">
        <xs:sequence minOccurs="1" maxOccurs="unbounded">
           <xs:element name="deviceGID" type="ObjectIdentifierList" minOccurs="0" maxOccurs="unbounded" />
           <xs:element name="validationDetails" type="ValidationDetails" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
      <xs:complexType name="ValidationDetails">
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
                  <xs:element name="title" type="xs:string" minOccurs="1"
                        maxOccurs="1" />
                  <xs:element name="severity" minOccurs="1" maxOccurs="1">
                        <xs:simpleType>
                              <xs:restriction base="xs:token">
                                    <xs:enumeration value="Critical" />
                                    <xs:enumeration value="Warning" />
                                    <xs:enumeration value="Info" />
                              </xs:restriction>
                        </xs:simpleType>
                  </xs:element>
                  <xs:element name="description" type="xs:string" minOccurs="1"
                        maxOccurs="1" />
                  <xs:element name="action" type="xs:string" minOccurs="0"
                        maxOccurs="1" />
            </xs:sequence>
      </xs:complexType>
            <xs:complexType name="DeploymentDeviceMsgDetails">
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
                  <xs:element name="Title" type="xs:string" minOccurs="1"
                        maxOccurs="1" />
                  <xs:element name="severityval" minOccurs="1" maxOccurs="1">
                        <xs:simpleType>
                              <xs:restriction base="xs:token">
                                    <xs:enumeration value="Critical" />
                                    <xs:enumeration value="Warning" />
                                    <xs:enumeration value="Info" />
                              </xs:restriction>
                        </xs:simpleType>
                  </xs:element>
                  <xs:element name="description" type="xs:string" minOccurs="1"
                        maxOccurs="1" />
                  <xs:element name="action" type="xs:string" minOccurs="0"
                        maxOccurs="1" />
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="DeploymentStatusDetails">
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
               <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1" />
```

```
                <xs:element name="deploymentMsgdetails" type="DeploymentDeviceMsgDetails" minOccurs="0"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DeploymentStatus">
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
              <xs:element name="deploymentstatus" type="DeploymentStatusDetails" minOccurs="0"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    <xs:element name="deploymentResponse" type="DeploymentResponse" />
        <xs:complexType name="DeploymentResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="status" type="xs:string"       minOccurs="1" maxOccurs="1" />
                        <xs:element name="deploymentJobGID" type="ObjectIdentifier" maxOccurs="1"
minOccurs="1"></xs:element>
                        <xs:element name="deploymentJobName" type="xs:string" minOccurs="1"
maxOccurs="1" />
                        <xs:element name="devicesInProgress" type="ObjectIdentifier" minOccurs="1"
maxOccurs="unbounded" />
                        <xs:element name="devicesSuccessList" type="DeploymentStatus" minOccurs="0"
maxOccurs="unbounded" />
                        <xs:element name="devicesFailureList" type="DeploymentStatus" minOccurs="0"
maxOccurs="unbounded" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="csmSessionOperationRequest" type="CSMSessionOperationRequest" />
        <xs:complexType name="CSMSessionOperationRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:choice id="select">
                            <xs:element name="csmSessionGID" type="ObjectIdentifier"
                                minOccurs="1" maxOccurs="1" />
                            <xs:element name="csmSessionName" type="xs:string" minOccurs="1"
                                maxOccurs="1" />
                        </xs:choice>
                        <xs:element name="comments" type="xs:string" minOccurs="0"
                            maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="csmApproveOperationRequest" type="CSMApproveOperationRequest" />
            <xs:complexType name="CSMApproveOperationRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:choice id="approveselect">
                            <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                            <xs:element name="csmSessionName" type="xs:string" minOccurs="1"
maxOccurs="1" />
                        </xs:choice>
                        <xs:element name="comments" type="xs:string" minOccurs="0" maxOccurs="1" />
```

```
                        <xs:element name="approvalStatus" type="xs:boolean"       minOccurs="1"
maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
            <xs:element name="csmApproveResultResponse" type="CSMApproveResultResponse" />
        <xs:complexType name="CSMApproveResultResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="result" type="xs:string" minOccurs="1"
                            maxOccurs="1" />
                        <xs:element name="csmSessionGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="submitCSMSessionRequest" type="SubmitCSMSessionRequest" />
        <xs:complexType name="SubmitCSMSessionRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                        <xs:element name="submitComments" type="xs:string"
                            minOccurs="0" maxOccurs="1" />
                        <xs:element name="continueOnWarnings" type="xs:boolean"
                            minOccurs="1" maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="csmSessionResultResponse" type="CSMSessionResultResponse" />
        <xs:complexType name="CSMSessionResultResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="result" type="xs:string" minOccurs="1"
                            maxOccurs="1" />
                        <xs:element name="csmSessionGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <xs:element name="addPolicyObjectRequest" type="ManagePolicyObjectRequest" />
        <xs:element name="modifyPolicyObjectRequest" type="ManagePolicyObjectRequest" />
        <xs:element name="deletePolicyObjectRequest" type="ManagePolicyObjectRequest" />
        <xs:element name="getPolicyObjectRequest" type="ManagePolicyObjectRequest" />
        <xs:complexType name="ManagePolicyObjectRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="0"
maxOccurs="1" />
                        <!-- if Redundant/Duplicate object is detected, throw warning (set: true) or continue (set:
false) with saving -->
```

```
                        <xs:element name="enforceDuplicateDetection" type="xs:boolean" minOccurs="0"
maxOccurs="1" />
                        <!-- ALL Supported POLICY OBJECT TYPES -->
                        <xs:element name="networkPolicyObject" type="NetworkPolicyObject" minOccurs="0"
maxOccurs="500" />
                        <xs:element name="portListPolicyObject" type="PortListPolicyObject" minOccurs="0"
maxOccurs="500" />
                        <xs:element name="servicePolicyObject" type="ServicePolicyObject" minOccurs="0"
maxOccurs="500" />
                        <xs:element name="interfaceRolePolicyObject" type="InterfaceRolePolicyObject"
minOccurs="0" maxOccurs="500" />
                        <xs:element name="timeRangePolicyObject" type="TimeRangePolicyObject"
minOccurs="0" maxOccurs="500" />
                        <xs:element name="identityUserGroupPolicyObject"
type="IdentityUserGroupPolicyObject" minOccurs="0" maxOccurs="500" />
                        <xs:element name="securityGroupPolicyObject" type="SecurityGroupPolicyObject"
minOccurs="0" maxOccurs="500" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <xs:element name="addPolicyObjectResponse" type="ManagePolicyObjectResponse" />
        <xs:element name="modifyPolicyObjectResponse" type="ManagePolicyObjectResponse" />
        <xs:element name="deletePolicyObjectResponse" type="ManagePolicyObjectResponse" />
        <xs:complexType name="ManagePolicyObjectResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="message" type="xs:string" minOccurs="0" maxOccurs="1" />
                        <xs:element name="resultObject" maxOccurs="unbounded" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element type="xs:string" name="objectGID" minOccurs="1"
maxOccurs="1" />
                                    <xs:element type="xs:string" name="name" minOccurs="1"
maxOccurs="1" />
                                    <xs:element type="xs:string" name="type" minOccurs="1" maxOccurs="1"
/>
                                    <xs:element type="xs:string" name="info" minOccurs="0" maxOccurs="1"
/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <xs:element name="policyObjectConfigByTypeRequest" type="PolicyObjectConfigByTypeRequest" />
        <xs:complexType name="PolicyObjectConfigByTypeRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="ObjectIdentifier"
                            minOccurs="0" maxOccurs="1" />
                        <xs:element name="policyObjectType" type="PolicyObjectType"
                            minOccurs="1" maxOccurs="1" />
                        <!-- if Redundant  object is detected do you want to throw warning or
                            continue with saving -->
```

```
                        <xs:element name="enforceConfDetection" type="xs:boolean"
                            minOccurs="1" maxOccurs="1" />
                        <!-- use deviceGID if you want to override the bb for already defined
                            global BB -->
                        <xs:element name="deviceGID" type="ObjectIdentifier"
                            minOccurs="0" maxOccurs="1" />
                        <xs:choice minOccurs="0" maxOccurs="1">
                            <!-- ALL Supported POLICY OBJECT TYPES -->
                            <xs:element name="networkPolicyObject" type="NetworkPolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <xs:element name="servicePolicyObject" type="ServicePolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <xs:element name="interfaceRolePolicyObject" type="InterfaceRolePolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <xs:element name="timeRangePolicyObject" type="TimeRangePolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <xs:element name="identityUserGroupPolicyObject"
type="IdentityUserGroupPolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <xs:element name="securityGroupPolicyObject" type="SecurityGroupPolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <xs:element name="portListPolicyObject" type="PortListPolicyObject"
                                minOccurs="0" maxOccurs="1" />
                            <!-- ..... all other policy objects .. -->

                        </xs:choice>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="deletePolicyObjectConfigByIdRequest" type="DeletePolicyObjectConfigByIdRequest" />
        <xs:complexType name="DeletePolicyObjectConfigByIdRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="xs:string" minOccurs="1"
                            maxOccurs="1" />
                        <xs:element name="objectGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                        <xs:element name="objectType" type="PolicyObjectType"
                            minOccurs="1" maxOccurs="1" />
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="policyObjectConfigResponse" type="PolicyObjectConfigResponse" />
        <xs:complexType name="PolicyObjectConfigResponse">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="objectGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1" />
                        <xs:element name="results" type="xs:string" minOccurs="0"
                            maxOccurs="1" />
                        <xs:element name="details" type="xs:string" minOccurs="0"
                            maxOccurs="1" />
                        <xs:element name="policyObject" minOccurs="1" maxOccurs="1">
                            <xs:complexType>
                                <xs:sequence minOccurs="1" maxOccurs="1">
                                    <!-- ALL DEFINED POLICY OBJECT TYPES -->
                                    <xs:element name="networkPolicyObject" type="NetworkPolicyObject"
```

```
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="portListPolicyObject" type="PortListPolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="servicePolicyObject" type="ServicePolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="interfaceRolePolicyObject"
type="InterfaceRolePolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="timeRangePolicyObject"
type="TimeRangePolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="slaMonitorPolicyObject"
type="SLAMonitorPolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="aclPolicyObject" type="ACLPolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="stdAcePolicyObject"
type="StandardACEPolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="extendedACEPolicyObject"
type="ExtendedACEPolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <xs:element name="identityUserGroupPolicyObject"
                                        type="IdentityUserGroupPolicyObject" minOccurs="0"
maxOccurs="unbounded" />
                                    <xs:element name="securityGroupPolicyObject"
type="SecurityGroupPolicyObject"
                                        minOccurs="0" maxOccurs="unbounded" />
                                    <!-- ..... all other policy objects .. -->
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    <xs:element name="overrideDetails" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="objectGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                                    <xs:element name="parentGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                                    <xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="updateNotesByRuleIDRequest" type="UpdateNotesByRuleIDRequest" />
        <xs:complexType name="UpdateNotesByRuleIDRequest">
            <xs:complexContent>
                <xs:extension base="BaseReqResp">
                    <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element name="csmSessionGID" type="ObjectIdentifier"
                            minOccurs="0" maxOccurs="1" />
                        <xs:element name="deviceGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                        <xs:element name="policyGID" type="ObjectIdentifier"
                            minOccurs="1" maxOccurs="1" />
                        <xs:element name="policyType" minOccurs="0" maxOccurs="1">
```

```
                                    <xs:simpleType>
                                        <xs:restriction base="xs:token">
                                            <xs:enumeration value="DeviceAccessRuleFirewallPolicy" />
                                            <xs:enumeration value="DeviceAccessRuleUnifiedFirewallPolicy" />
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="description" type="xs:string"
                                    minOccurs="1" maxOccurs="1" />
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
                <xs:element name="reorderRulesRequest" type="ReorderRulesRequest" />
                <xs:complexType name="ReorderRulesRequest">
                    <xs:complexContent>
                        <xs:extension base="BaseReqResp">
                            <xs:sequence minOccurs="1" maxOccurs="1">
                                <xs:element name="csmSessionGID" type="ObjectIdentifier"
                                    minOccurs="0" maxOccurs="1" />
                                <xs:choice minOccurs="1" maxOccurs="1">
                                    <xs:element name="deviceGID" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />
                                    <xs:element name="name" type="xs:string"
                                minOccurs="1" maxOccurs="1" />
                                </xs:choice>
                                <xs:element name="ruleGID" type="ObjectIdentifier"
                                    minOccurs="1" maxOccurs="1" />

                                <xs:element name="policyType" type="PolicyConfigType" minOccurs="0"
maxOccurs="1"/>
                                <xs:element name="toOrderId" type="xs:unsignedInt"
                                    minOccurs="1" maxOccurs="1" />
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>


                <xs:complexType name="DeployableDevicesListRequest">
                    <xs:complexContent>
                        <xs:extension base="BaseReqResp">
                            <xs:sequence minOccurs="1" maxOccurs="1">
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
                <xs:element name="deployableDevicesListRequest" type="DeployableDevicesListRequest"></xs:element>
                <xs:complexType name="DeployJobStatusRequest">
                    <xs:complexContent>
                        <xs:extension base="BaseReqResp">
                            <xs:sequence minOccurs="1" maxOccurs="1">
                              <xs:choice id="selectReqType">
                              <xs:element name="deploymentJobName" type="xs:string" minOccurs="1" maxOccurs="1" />
                                <xs:element name="deploymentJobGID" type="ObjectIdentifier" minOccurs="1"
maxOccurs="1" />
                                </xs:choice>
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
```

```
</xs:complexType>
<xs:element name="deployJobStatusRequest" type="DeployJobStatusRequest"></xs:element>

<xs:complexType name="DeployableDevicesListResponse">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="deviceGIDs" type="ObjectIdentifierList"
                    minOccurs="0" maxOccurs="1" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:element name="getPolicyObjectByGID" type="GetPolicyObjectByGID" />
<xs:complexType name="GetPolicyObjectByGID">
    <xs:complexContent>
        <xs:extension base="BaseReqResp">
            <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="csmSessionGID" type="ObjectIdentifier" minOccurs="0" maxOccurs="1" />

            <xs:element name="gid" type="ObjectIdentifier" minOccurs="1" maxOccurs="unbounded"/>

            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:simpleType name="PolicyObjectType">

    <xs:restriction base="xs:token">
        <xs:enumeration value="NetworkPolicyObject" />
        <xs:enumeration value="ServicePolicyObject" />
        <xs:enumeration value="IdentityUserGroupPolicyObject" />
        <xs:enumeration value="InterfaceRolePolicyObject" />
        <xs:enumeration value="SecurityGroupPolicyObject" />
        <xs:enumeration value="TimeRangePolicyObject" />
        <xs:enumeration value="PortListPolicyObject" />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PolicyObjectTypeExtended">
    <xs:restriction base="xs:token">
        <xs:enumeration value="NetworkPolicyObject" />
        <xs:enumeration value="ServicePolicyObject" />
        <xs:enumeration value="IdentityUserGroupPolicyObject" />
        <xs:enumeration value="InterfaceRolePolicyObject" />
        <xs:enumeration value="SecurityGroupPolicyObject" />
        <xs:enumeration value="TimeRangePolicyObject" />
        <xs:enumeration value="StandardACEPolicyObject" />
        <xs:enumeration value="ExtendedACEPolicyObject" />
        <xs:enumeration value="ACLPolicyObject" />
        <xs:enumeration value="PortListPolicyObject" />
        <xs:enumeration value="SLAMonitorPolicyObject" />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="OperationStatus">
    <xs:restriction base="xs:token">
        <xs:enumeration value="SUCCESS" />
        <xs:enumeration value="FAILED" />
    </xs:restriction>
```

```
    </xs:simpleType>
<xs:complexType name="FirewallACLSettingsPolicy">
    <xs:complexContent>
        <xs:extension base="BasePolicy">
            <xs:sequence minOccurs="1" maxOccurs="1">
                <!-- aclName is mandatory if request has Global interface or if user
                    defined acl is set as true. if useUserDefinedACLName defined aclname is true
                    then aclName will be ignored -->
                <xs:element name="aclName" type="xs:string" minOccurs="0"
                    maxOccurs="1" />
                <xs:element name="interfaceGID" type="ObjectIdentifier"
                    minOccurs="0" maxOccurs="1" />
                <xs:element name="trafficDirection" minOccurs="1"
                    maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="In" />
                            <xs:enumeration value="Out" />
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="useUserDefinedACLName" type="xs:boolean"
                    minOccurs="1" maxOccurs="1" />
                <xs:element name="enablePerUserDownloadableACLs" type="xs:boolean"
                    minOccurs="1" maxOccurs="1" />
                <xs:element name="enableObjectGroupSearch" type="xs:boolean"
                    minOccurs="1" maxOccurs="1" />
                <xs:element name="enableAccessListCompilation" type="xs:boolean"
                    minOccurs="0" maxOccurs="1" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

    <xs:simpleType name="PolicyConfigType">
        <xs:restriction base="xs:token">
            <xs:enumeration value="DeviceAccessRuleFirewallPolicy" />
            <xs:enumeration value="DeviceAccessRuleUnifiedFirewallPolicy" />
            <xs:enumeration value="FirewallACLSettingsPolicy" />
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="policyObjectsListByTypeRequest" type="PolicyObjectsListByTypeRequest" />
    <xs:complexType name="PolicyObjectsListByTypeRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="policyObjectType" type="PolicyObjectType"
                        minOccurs="1" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="PolicyObjectsListResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence minOccurs="1" maxOccurs="1">

                    <xs:element name="policyObject" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence minOccurs="1" maxOccurs="1">
```

```
                                        <!-- ALL DEFINED POLICY OBJECT TYPES -->
                                        <xs:element name="networkPolicyObject" type="NetworkPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="portListPolicyObject" type="PortListPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="servicePolicyObject" type="ServicePolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="interfaceRolePolicyObject"
type="InterfaceRolePolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="timeRangePolicyObject"
type="TimeRangePolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="slaMonitorPolicyObject"
type="SLAMonitorPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="aclPolicyObject" type="ACLPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="stdAcePolicyObject"
type="StandardACEPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="extendedACEPolicyObject"
type="ExtendedACEPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <xs:element name="identityUserGroupPolicyObject"
                                            type="IdentityUserGroupPolicyObject" minOccurs="0"
maxOccurs="unbounded" />
                                        <xs:element name="securityGroupPolicyObject"
type="SecurityGroupPolicyObject"
                                            minOccurs="0" maxOccurs="unbounded" />
                                        <!-- ..... all other policy objects .. -->
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
</xs:schema>
```

11.3 Event XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="csm" targetNamespace="csm">
	<xs:include schemaLocation="common.xsd"/>
	<xs:simpleType name="EventType">
		<xs:restriction base="xs:token">
			<xs:enumeration value="syslog"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="EventFormat">
		<xs:restriction base="xs:token">
			<xs:enumeration value="xml"/>
			<xs:enumeration value="plainText"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="EventCategory">
		<xs:restriction base="xs:token">
			<xs:enumeration value="configChange"/>
			<xs:enumeration value="deviceStatus"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:element name="event" type="Event"/>
	<xs:complexType name="Event">
		<xs:choice>
			<xs:element name="configChange" type="ConfigChangeEvent"/>
		</xs:choice>
	</xs:complexType>
	<xs:simpleType name="UpdateType">
		<xs:restriction base="xs:token">
			<xs:enumeration value="NO_OOB"/>
			<xs:enumeration value="OOB"/>
			<xs:enumeration value="DEVICE_DOWN"/>
			<xs:enumeration value="DEVICE_UP"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="DeploymentType">
		<xs:restriction base="xs:token">
			<xs:enumeration value="Device"/>
			<xs:enumeration value="File"/>
			<xs:enumeration value="AUS"/>
			<xs:enumeration value="CNS"/>
			<xs:enumeration value="TMS"/>
			<xs:enumeration value="Unknown"/>
			<xs:enumeration value="NOT_APPLICABLE"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="BaseEventDetails">
		<xs:sequence>
			<xs:element name="subscriptionId" type="xs:string" minOccurs="1" maxOccurs="1"/>
			<xs:element name="eventType" type="EventType" minOccurs="1" maxOccurs="1"/>
			<xs:element name="eventCategory" type="EventCategory" minOccurs="1" maxOccurs="1"/>
			<xs:element name="time" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
			<xs:element name="content" type="xs:string" minOccurs="1" maxOccurs="1"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DeviceSpecificEvent">
		<xs:complexContent>
			<xs:extension base="BaseEventDetails">
				<xs:sequence>
					<xs:element name="srcIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
					<xs:element name="srcGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
					<xs:element name="srcDns" type="xs:string" minOccurs="0" maxOccurs="1"/>
					<xs:element name="srcOSType" type="OSType" minOccurs="1" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
```

```
    </xs:complexType>
    <!-- Config Change Event -->
    <xs:element name="configChangeEvent" type="ConfigChangeEvent"/>
    <xs:complexType name="ConfigChangeEvent">
        <xs:complexContent>
            <xs:extension base="DeviceSpecificEvent">
                <xs:sequence>
                    <xs:element name="deploymentType" type="DeploymentType" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="updateType" type="UpdateType" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="deviceStatusEvent" type="DeviceStatusEvent"/>
    <xs:complexType name="DeviceStatusEvent">
        <xs:complexContent>
            <xs:extension base="DeviceSpecificEvent">
                <xs:sequence>
                    <xs:element name="updateType" type="UpdateType" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="EventFilterItem">
        <xs:sequence>
            <xs:element name="filterEventType" type="EventType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="filterEventFormat" type="EventFormat" minOccurs="1" maxOccurs="1"/>
            <xs:element name="filterEventCategory" type="EventCategory" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SyslogServer">
        <xs:sequence>
            <xs:element name="port" type="xs:positiveInteger" default="514" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destAddress" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="eventSubRequest" type="EventSubRequest"/>
    <xs:complexType name="EventSubRequest">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence>
                    <xs:element name="op" type="SubscriptionOperation" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="subscriptionId" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="eventFilterItem" type="EventFilterItem" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="syslogServer" type="SyslogServer" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="eventSubResponse" type="EventSubResponse"/>
    <xs:complexType name="EventSubResponse">
        <xs:complexContent>
            <xs:extension base="BaseReqResp">
                <xs:sequence>
                    <xs:element name="subscriptionId" type="xs:string" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

11.4 Utility XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2011 (http://www.altova.com) by BRIAN MCMAHON (CISCO) -->
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="csm" targetNamespace="csm">
	<xs:include schemaLocation="common.xsd"/>
	<xs:simpleType name="Result">
		<xs:restriction base="xs:token">
			<xs:enumeration value="ok"/>
			<xs:enumeration value="timeout"/>
			<xs:enumeration value="failed"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="DeviceReadOnlyCLICmd">
		<xs:sequence>
			<xs:choice minOccurs="1" maxOccurs="1">
				<xs:element name="deviceIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
				<xs:element name="deviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
				<xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
			</xs:choice>
			<xs:element name="cmd" minOccurs="1" maxOccurs="1">
				<xs:simpleType>
					<xs:restriction base="xs:string">
						<xs:pattern value="[sS][hH][oO][wW]"/>
					</xs:restriction>
				</xs:simpleType>
			</xs:element>
			<xs:element name="argument" type="xs:string" minOccurs="1" maxOccurs="1"/>
			<xs:element name="execTimeout" type="xs:unsignedInt" minOccurs="0" maxOccurs="1">
				<xs:simpleType>
					<xs:restriction base="xs:unsignedInt">
						<xs:minInclusive value="1"/>
					</xs:restriction>
				</xs:simpleType>
		  </xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DeviceCmdResult">
		<xs:sequence>
			<xs:element name="deviceIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
			<xs:element name="deviceGID" type="ObjectIdentifier" minOccurs="1" maxOccurs="1"/>
			<xs:element name="deviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
			<xs:element name="result" type="Result" minOccurs="1" maxOccurs="1"/>
			<xs:element name="resultContent" type="xs:string" minOccurs="0" maxOccurs="1"/>
		</xs:sequence>
	</xs:complexType>
	<xs:element name="execDeviceReadOnlyCLICmdsRequest" type="ExecDeviceReadOnlyCLICmdsRequest"/>
	<xs:complexType name="ExecDeviceReadOnlyCLICmdsRequest">
		<xs:complexContent>
			<xs:extension base="BaseReqResp">
				<xs:sequence>
					<xs:element name="deviceReadOnlyCLICmd" type="DeviceReadOnlyCLICmd" minOccurs="1"
maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="execDeviceReadOnlyCLICmdsResponse" type="ExecDeviceReadOnlyCLICmdsResponse"/>
	<xs:complexType name="ExecDeviceReadOnlyCLICmdsResponse">
		<xs:complexContent>
			<xs:extension base="BaseReqResp">
				<xs:sequence>
					<xs:element name="deviceCmdResult" type="DeviceCmdResult" minOccurs="1" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
</xs:schema>
```

End of Document