



Send document comments to vnmc-docfeedback@cisco.com.



Cisco Virtual Network Management Center XML API Reference Guide, Release 1.0.1

January 24, 2011

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-24045-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco Virtual Network Management Center XML API Reference Guide, Release 1.0.1
© 2011 Cisco Systems, Inc. All rights reserved.

Send document comments to vnmc-docfeedback@cisco.com.



CONTENTS

Preface	ix	
Audience	ix	
Organization	ix	
Document Conventions	x	
Related Documentation	xi	
Cisco Virtual Network Management Center Documentation	xi	
Cisco Virtual Security Gateway Documentation	xi	
Cisco Nexus 1000V Series Switch Documentation	xii	
Obtaining Documentation and Submitting a Service Request	xi	
CHAPTER 1	Introduction to Cisco VNM XML API Reference Guide	1-1
	Information About Cisco VNM and the XML API	1-1
	Overview of Cisco VNM	1-1
	VNM Management Information Model	1-2
	Description of VNM Components	1-2
	Management Controller	1-2
	Service Registry	1-3
	Resource Manager	1-3
	Policy Manager	1-4
	VM Manager	1-4
	Overview of the VNM XML API	1-4
	VNM Data Model Schema	1-5
	Access VNM Services	1-6
	Object Naming	1-6
	Distinguished Name	1-6
	Relative Name	1-6
	API Method Categories	1-6
	Authentication Methods	1-7
	Query Methods	1-7
	Query Filters	1-8
	Configuration Methods	1-9
	Event Subscription Methods	1-9
	Capturing XML Interchange between the GUI and VNM Server	1-10
	Success or Failure Responses	1-10

Send document comments to vnmc-docfeedback@cisco.com.

Successful Response	1-10
Failed Requests	1-11
Empty Results	1-11

CHAPTER 2

Using the VNMC XML API 2-1

Tutorial Examples	2-1
Authentication Methods	2-1
Login	2-1
Query Methods	2-3
Using configResolveDn	2-3
Using configResolveDns	2-3
Using configResolveClass	2-4
Using configResolveClasses	2-4
Using configFindDnsByClassId	2-4
Using configResolveChildren	2-5
Using configResolveParent	2-5
Using configScope	2-5
Using Query Methods for Policies	2-6
Querying Faults	2-6
Using Filters	2-7
Simple Filters	2-7
Property Filters	2-7
Composite Filters	2-10
Modifier Filter	2-11
VNMC XML API Use Case Example	2-12
Authentication via Management Controller	2-12
Login	2-12
Tenant Management via Service Registry	2-13
Policy Management	2-13
Device Policy	2-14
Device Profile	2-17
Zone	2-18
Object Group	2-19
Attribute Dictionary	2-20
Policy	2-20
PolicySet	2-22
Security Profile	2-22
Resource Management	2-23
Compute Firewall	2-23

Send document comments to vnmc-docfeedback@cisco.com.

Query Firewall Instances	2-24
Associate Firewall Instance	2-25

CHAPTER 3
VNMC XML API Method Descriptions 3-1

VNMC XML API Methods	3-1
aaaGetRemoteUserRoles	3-1
Request syntax	3-1
Response syntax	3-2
Examples	3-2
aaaGetUserLocales	3-2
Request syntax	3-3
Response syntax	3-3
Examples	3-3
aaaKeepAlive	3-4
Request syntax	3-4
Response syntax	3-4
Examples	3-4
aaaLogin	3-5
Request syntax	3-5
Response syntax	3-5
Examples	3-6
aaaLogout	3-6
Request syntax	3-6
Response syntax	3-7
Examples	3-7
aaaRefresh	3-7
Request syntax	3-7
Response syntax	3-8
Examples	3-8
configConfFiltered	3-9
Request syntax	3-9
Response syntax	3-9
Examples	3-10
configConfMo	3-10
Request syntax	3-11
Response syntax	3-11
Examples	3-11
configConfMoGroup	3-12
Request syntax	3-12
Response syntax	3-13

Send document comments to vnmc-docfeedback@cisco.com.

Examples	3-13
configConfMos	3-14
Request syntax	3-14
Response syntax	3-14
Examples	3-15
configFindDnsByClassId	3-16
Request syntax	3-16
Response syntax	3-16
Examples	3-16
configMoChangeEvent	3-17
Request syntax	3-17
Response syntax	3-17
Examples	3-17
configResolveChildren	3-18
Request syntax	3-18
Response syntax	3-18
Examples	3-19
configResolveClass	3-20
Request syntax	3-20
Response syntax	3-20
Examples	3-20
configResolveClasses	3-21
Request syntax	3-21
Response syntax	3-21
Examples	3-22
configResolveDn	3-22
Request syntax	3-23
Response syntax	3-23
Examples	3-23
configResolveDns	3-24
Request syntax	3-24
Response syntax	3-24
Examples	3-25
configResolveParent	3-26
Request syntax	3-26
Response syntax	3-26
Examples	3-27
configScope	3-28
Request syntax	3-28
Response syntax	3-28

Send document comments to vnmc-docfeedback@cisco.com.

Examples	3-29
eventSendHeartbeat	3-29
Request syntax	3-29
Response syntax	3-30
Examples	3-30
eventSubscribe	3-30
Request syntax	3-30
Response syntax	3-30
Examples	3-31
eventSubscribeApps	3-31
Request syntax	3-31
Response syntax	3-31
Examples	3-31
faultAckFault	3-32
Request syntax	3-32
Response syntax	3-32
Examples	3-32
faultAckFaults	3-33
Request syntax	3-33
Response syntax	3-33
Examples	3-33
faultResolveFault	3-34
Request syntax	3-34
Response syntax	3-34
Examples	3-34
loggingSyncOcn	3-35
Request syntax	3-35
Response syntax	3-35
Examples	3-35
methodVessel	3-36
Request syntax	3-36
Response syntax	3-36
Examples	3-36
orgResolveElements	3-39
Request syntax	3-39
Response syntax	3-40
Examples	3-40
orgResolveInScope	3-41
Request syntax	3-42
Response syntax	3-42

Send document comments to vnmc-docfeedback@cisco.com.

Examples	3-43
orgResolveLogicalParents	3-44
Request syntax	3-44
Response syntax	3-44
Examples	3-45
policyEstimateImpact	3-45
Request syntax	3-45
Response syntax	3-46
Examples	3-46
poolResolveInScope	3-47
Request syntax	3-47
Response syntax	3-48
Examples	3-48

Send document comments to vnmc-docfeedback@cisco.com.



Preface

This preface describes the audience, organization, and conventions of the *Cisco Virtual Network Management Center XML API Reference Guide, Release 1.0.1*. It also provides information on how to obtain related documentation.

This chapter includes the following sections:

- [Audience, page ix](#)
- [Organization, page ix](#)
- [Document Conventions, page x](#)
- [Related Documentation, page xi](#)
- [Obtaining Documentation and Submitting a Service Request, page xii](#)

Audience

This guide is intended for software engineers with a background in programming and the use of application programming interfaces (APIs). Engineers should have knowledge of XML, data systems, networking protocols, and storage protocols.

Organization

This guide is organized as follows:

Chapter and Title	Description
Chapter 1, “Introduction to VNMC XML API Reference Guide”	Provides an overview of the VNMC XML API.
Chapter 2, “Using the VNMC XML API”	Provides a tutorial and details on a use case example.
Chapter 3, “API Method Descriptions”	Descriptions of the API method with syntax and working examples.

[Send document comments to vnmc-docfeedback@cisco.com.](mailto:vnmc-docfeedback@cisco.com)

Document Conventions

This document uses the following conventions:



Note

Means reader *take note*. Notes contain helpful suggestions or references to material not covered in the manual.



Caution

Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.



Tip

Means *the following information will help you solve a problem*.

Command descriptions use these conventions:

Convention	Description
boldface font	Commands and keywords are in boldface.
italic font	Parameters for which you supply values are in italics.
[]	Elements in square brackets are optional.
[x y z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.

Screen examples use these conventions:

screen font	Terminal sessions and information that the switch displays are in screen font.
boldface screen font	Information that you must enter is in boldface screen font.
<i>italic screen font</i>	Arguments for which you supply values are in italic screen font.
<>	Non-printing characters, such as passwords, are in angle brackets.
[]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or number sign (#) at the beginning of a line of code indicates a comment line.

Send document comments to vnmc-docfeedback@cisco.com.

Related Documentation

This section contains information about the documentation available for Cisco Virtual Network Management Center and related products.

This section includes the following topics:

- [Cisco Virtual Network Management Center Documentation](#), page xi
- [Cisco Virtual Security Gateway Documentation](#), page xi
- [Cisco Nexus 1000V Series Switch Documentation](#), page xii

Cisco Virtual Network Management Center Documentation

The following Cisco Virtual Network Management Center documents are available on Cisco.com at the following URL:

http://www.cisco.com/en/US/products/ps11213/tsd_products_support_series_home.html

- *Release Notes for Cisco Virtual Network Management Center, Release 1.0.1*
- *Cisco Virtual Security Gateway, Release 4.2(1)VSG1(1) and Cisco Virtual Network Management Center, Release 1.0.1 Installation Guide*
- *Cisco Virtual Network Management Center CLI Configuration Guide, Release 1.0.1*
- *Cisco Virtual Network Management Center GUI Configuration Guide, Release 1.0.1*
- *Cisco Virtual Network Management Center XML API Reference Guide, Release 1.0.1*

Cisco Virtual Security Gateway Documentation

The following Cisco Virtual Security Gateway for the Nexus 1000V Series Switch documents are available on Cisco.com at the following url:

http://www.cisco.com/en/US/products/ps11208/tsd_products_support_model_home.html

- *Cisco Virtual Security Gateway for Nexus 1000V Series Switch Release Notes, Release 4.2(1)VSG(1)*
- *Cisco Virtual Security Gateway, Release 4.2(1)VSG1(1) and Cisco Virtual Network Management Center, Release 1.0.1 Installation Guide*
- *Cisco Virtual Security Gateway for Nexus 1000V Series Switch License Configuration Guide, Release 4.2(1)VSG1(1)*
- *Cisco Virtual Security Gateway for Nexus 1000V Series Switch Configuration Guide, Release 4.2(1)VSG1(1)*
- *Cisco Virtual Security Gateway for Nexus 1000V Series Switch Command Reference, Release 4.2(1)VSG1(1)*
- *Cisco Virtual Security Gateway for Nexus 1000V Series Switch Troubleshooting Guide, Release 4.2(1)VSG(1)*

Send document comments to vnmc-docfeedback@cisco.com.

Cisco Nexus 1000V Series Switch Documentation

The Cisco Nexus 1000V Series switch documentation is available at the following URL:

http://www.cisco.com/en/US/products/ps9902/tsd_products_support_series_home.html

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

<http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html>

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service. Cisco currently supports RSS Version 2.0.

This document is to be used in conjunction with the documents listed in the “[Related Documentation](#)” section.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

© 2011 Cisco Systems, Inc. All rights reserved.



CHAPTER 1

Introduction to Cisco VNMC XML API Reference Guide

This chapter provides general information about the XML Application Programming Interface (API). The following sections are:

- [Information About Cisco VNMC and the XML API, page 1-1](#)
- [API Method Categories, page 1-6](#)
- [Capturing XML Interchange between the GUI and VNMC Server, page 1-10](#)
- [Success or Failure Responses, page 1-10](#)

Information About Cisco VNMC and the XML API

This section contains the following topics:

- [Overview of Cisco VNMC, page 1-1](#)
- [VNMC Management Information Model, page 1-2](#)
- [Description of VNMC Components, page 1-2](#)
- [VNMC Data Model Schema, page 1-5](#)
- [VNMC Data Model Schema, page 1-5](#)
- [Access VNMC Services, page 1-6](#)
- [Object Naming, page 1-6](#)
- [Authentication Methods, page 1-7](#)

Overview of Cisco VNMC

Cisco Virtual Network Management Center (VNMC) is a virtual appliance that provides centralized device and security policy management for the Cisco Virtual Security Gateway (VSG) and the Cisco Nexus 1000V series switches. Designed for multitenant operation, VNMC provides seamless, scalable, and automation-centric management for virtualized data center and cloud environments. With a web-based GUI, CLI, and XML API methods, VNMC allows you to manage VSGs that are deployed throughout the data center from a centralized location.

Send document comments to vnmc-docfeedback@cisco.com.

Multitenancy refers to an architecture principle where multiple client organizations or tenants can have their own virtualized compute, network and storage resources, which are deployed over a shared physical infrastructure. While multiple tenants may coexist on the same infrastructure, each tenant can have administrative privileges for its virtualized resources. The system is designed to meet the specified SLA for each tenant, including compute, network, storage and security policies.

VNMC is built on an information model-driven architecture, where each managed device is represented by its subcomponents. This architecture enables VNMC to provide greater agility and simplification for securing multitenant infrastructure.

VNMC Management Information Model

All the physical and logical components that comprise a Cisco VNMC service component are represented in a hierarchical management information model. This model is referred to as the management information tree (MIT). Each node in the tree represents a managed object (or group of objects) that contains its administrative state and its operational state. The hierarchical structure starts at the top and contains parent and child nodes. Each node in this tree is a managed object and each object has a unique distinguished name (DN) that describes the object and its place in the tree.

Managed objects are abstractions of VNMC-managed entities such as policies, rules, security profiles and VSG instances. Certain managed objects are not created by users but are automatically created by VNMC. By invoking the API, objects are read from and written to the MIT. The information model of an individual VNMC service component is centrally stored and managed by its data management engine (DME). When a user initiates an administrative change to a VNMC service component (for example, associating a compute firewall profile to a VSG), the DME first applies that change to the information model, and later the change is applied to the actual VSG. This approach is called a model-driven framework.

Description of VNMC Components

VNMC consists of multiple service components to provide modularized functionalities for management, tenant management, policy management, resource management, and so on. These components are also called service providers or applications. Each of the components are accessed through a unique URL. They are as follows:

- [Management Controller, page 1-2](#)
- [Service Registry, page 1-3](#)
- [Resource Manager, page 1-3](#)
- [Policy Manager, page 1-4](#)
- [VM Manager, page 1-4](#)

Each component has its own data model and a DME (data model engine) to process the model-driven service requests. Each component maintains its own management information tree storage (both in memory and in the persistence store). Data sharing across different service components are achieved with ad-hoc inter-service API communications or the publish/subscribe mechanism.

Management Controller

The Management Controller (also known as core service) provides the system-related service for the VNMC virtual machine. Its services include the following:

Send document comments to vnmc-docfeedback@cisco.com.

- Manages user login authentications and authorizations in local or LDAP mode
- Management of access controls such as locales, roles, and trusted points, and so on
- Administers system information such as the IP address, subnet mask, gateway, hostname, and so on.
- Runs system maintenance such as database backup, data export/import, and so on.
- Maintains system diagnostic information such as audit logs, faults, event logs and core dump files

The Management Controller's type is "mgmt-controller". Use this service type in the API URL for all Management Controller-related requests

Service Registry

VNMC is designed with a distributed architecture where Service Registry is the central service repository which holds information about all registered managed end-points (VSM, VSG) and the service providers (Policy Manager, Resource Manager, and so on.).

Service endpoints and the service providers register themselves dynamically with the service registry and retrieve the information about interested service components from Service Registry. Service Registry is also responsible for tenant management. Its services include the following:

- Creates, deletes and updates organizations (tenants, data centers, vApps, tiers). Organization changes are automatically propagated to the Policy Manger and Resource Manager where policies and resources are attached to the intended organizations, respectively.
- Maintains information about the registered services (providers, endpoints, management controller, service registry)
- Maintains diagnostic information, such as audit logs, faults, event logs.

The service registry's type is "service-reg". Use this service type in the API URL for all Service Registry-related requests.

Resource Manager

The Resource Manager manages logical compute firewalls and their association with actual VSGs. When a compute firewall is associated with a VSG, the device configuration profile information (defined by the compute firewall) is pushed to the actual VSG. This triggers the VSG to download the security profiles and policies from the Policy Manager. Services include the following:

- Maintains inventory of Virtual Security Gateways (VSG) and Virtual Supervisor Modules (VSM)
- Defines compute firewalls and associates them with VSGs for provisioning
- Integrates with VMWare vCenter instances to retrieve VM attributes
- Maintains inventory of discovered Virtual Machines and distributes them to VSG instances. This includes:
 - VM attributes (name, hypervisor, parent vApp, cluster)
 - vNIC attributes (port profile name, IP addresses)
- Manages pools of VSGs
- Maintans health states and faults for VSGs
- Maintains diagnostic information, such as audit logs, faults, event logs

The Resource Manager's type is "resource-mgr". Use this service type in the API URL for all Resource Manager-related requests.

Send document comments to vnmc-docfeedback@cisco.com.

Policy Manager

The Policy Manager is the central repository of device configuration profiles, security policies and all associated artifacts. When a compute firewall is associated with a VSG from the Resource Manager, the VSG queries the Policy Manager to resolve the device profile, security profiles, and all referenced policies. The VSG then configures itself according to the information retrieved from the Policy Manager. Services include the following:

- Defines policies
- Defines policy sets and assigns policies
- Defines policy rules with conditions on
 - Network attributes (i.e., protocol, source/destination IP address and port)
 - VM attributes (instance name, guest OS, zone, parent app, port profile, cluster, hypervisor)
 - Custom attributes
- Defines zones
- Defines object groups
- Defines the security profile dictionary and custom attributes
- Defines security profiles and assign policy sets
- Defines firewall device profiles
- Defines VNMC system management device profile and policies for NTP, DNS, syslog and faults
- Distributes policies, security profiles, device profiles and associated objects to VSG instances
- Maintains diagnostic information, such as audit logs, faults, event logs

The Policy Manager's service type is "policy-mgr". Use this service type in the API URL for all Policy Manager-related requests.

VM Manager

VM Manager is responsible for interacting with VMWare vCenter and maintaining the VM information retrieved from vCenter. It is a backend service without any user-accessible services. The VM Manager's service type is 'vm-manager'.

Overview of the VNMC XML API

The VNMC XML API is a programmatic way to integrate or interact with the Cisco Virtual Network Management Center. The API interface accepts XML documents via the HTTPS protocol. Developers can use any programming language to generate XML documents that contain the API methods. Configuration and state information is stored in a hierarchical tree structure known as the management information tree (completely exposed through the XML API).

The API model is recursively driven and provides major functionality for application development. For example, changes can be made on a single object, an object subtree, or the entire object tree. Changes can be made to a single attribute on an object or applied to the entire VNMC structure with a single API call.

Send document comments to vnmc-docfeedback@cisco.com.

The API operates in “forgiving mode.” Missing attributes are substituted with default values (if applicable) that are maintained in the internal data management engine (DME). If configuring multiple managed objects (that is, policies), and any of the managed objects cannot be configured, the API stops its operation, returns the configuration to its prior state, and then stops the API process with a fault notification.

The API leverages an asynchronous operations model to improve scalability and performance. Processes that require time to complete are nonblocking (faster API processes can proceed). A process receives a success message upon a valid request and a complete message when the task is finished.

Full event subscription is supported. VNMC sends notifications to all the subscribers for the events (for example, changes to managed objects) that occur, and indicates the type of state change.

Future updates to the managed object data model will conform to the existing object model to ensure backward compatibility. If existing properties are changed during a product upgrade, it will be handled during the database load after the upgrade. New properties will be assigned default values.

VNMC uses a model-driven architecture, where changes are first applied to logical constructs in the form of managed objects. The managed objects then apply the changes to the endpoints to achieve the required state (configuration) of the endpoint.

Operation of the API is transactional and terminates on a single data model. VNMC is responsible for all endpoint (VSG, VSM) communication (such as state updates). Users cannot communicate directly to endpoints, which relieves developers from the task of administering isolated, individual component configurations.

The VNMC API model includes the following programmatic entities:

- Classes—Properties and states of objects in the management information tree
- Methods—Actions that the API performs on one or more objects
- Types—Object properties that map values to the object state (that is, policyDeviceProfile)

A typical request comes into a VNMC service component’s DME and is placed in the transactor queue in FIFO fashion. The transactor gets the request from the queue, interprets the request, and performs an authorization check. When the request is confirmed, the transactor updates the management information tree. This process is done in a single transaction.

VNMC Data Model Schema

The data model schema files for all VNMC service providers are packaged in the VNMC server and can be retrieved using the URL:

```
https://<vnmc ip address>/schema
```

The schema list includes:

- core.in.xsd—Management controller configuration APIs and data model
- core.out.xsd—Management controller query APIs and data model
- policy-mgr.in.xsd—Policy manager configuration APIs and data model
- policy-mgr.out.xsd—Policy manager query APIs and data model
- resource-mgr.in.xsd—Resource manager configuration APIs and data model
- resource-mgr.out.xsd—Resource manager query APIs and data model
- service-reg.in.xsd—Service registry configuration APIs and data model
- service-reg.out.xsd—Service registry query APIs and data model

Send document comments to vnmc-docfeedback@cisco.com.

Access VNMC Services

Access using XML API request over HTTPS protocol. The HTTPS request URL format is:

```
https://<vnmc ip address>/xmlIM/<service type>
```

where <service type> is the service type of the intended service provider as described in the VNMC Components section. For example, to submit a request to the Policy Manager, use service type "policy-mgr" in the URL:

```
https://<vnmc ip address>/xmlIM/policy-mgr
```

Object Naming

You can identify a specific object by its distinguished name (DN) or by its relative name (RN).

This section contains the following topics:

- [Distinguished Name, page 1-6](#)
- [Relative Name, page 1-6](#)

Distinguished Name

The DN enables you to unambiguously identify a target object. The DN has the following format consisting of a series of relative names:

```
dn = {rn}/{rn}/{rn}/{rn}...
```

Example: org-root/org-Cisco/zone-trustedServers-0

In the above-mentioned example, the DN provides a fully qualified path for a Zone object zone-trustedServers-0 from the top of the object tree to the object. The DN specifies the exact managed object on which the API call is operating.

```
< ... dn = "org-root/org-Cisco/zone-trustedServers-0" />
```

Relative Name

The relative name (RN) identifies an object within the context of its parent object. The DN of an object is comprised of its parent's DN and its RN in the format of:

```
dn = <parent dn>/<rn>
```

For example, for a Zone object with the name "trustedServers-0" defined under the tenant "Cisco", its rn is "zone-trustedServers-0". Its parent tenant DN is "org-root/org-Cisco". Its DN is "org-root/org-Cisco/zone-trustedServers-0".

API Method Categories

There are four categories of methods used to interact with the VNMC. Each API is a method, and each method corresponds to an XML document. The methods are:

- [Authentication Methods, page 1-7](#)
- [Query Methods, page 1-7](#)

Send document comments to vnmc-docfeedback@cisco.com.

- [Configuration Methods, page 1-9](#)
- [Event Subscription Methods, page 1-9](#)



Note

Several code examples in this guide substitute the term “<real_cookie>” for an actual cookie (such as 1217377205/85f7ff49-e4ec-42fc-9437-da77a1a2c4bf). The VNMC cookie is a 47-character string; it is not the type of cookie that web browsers store locally to maintain session information.

Authentication Methods

Authentication methods authenticate and maintain the session. For example,

- **aaaLogin**—Initial method for logging in to VNMC
- **aaaRefresh**—Refreshes the current authentication cookie
- **aaaLogout**—Exits the current session and deactivates the current authentication cookie

Authentication methods initiate and maintain an active VNMC session. A successful authentication must be performed before other API calls are allowed. API requests are cookie authenticated.

After a connection session is established and authenticated, a cookie is returned in the response. It is valid for 7200 seconds (120 minutes). The cookie must be refreshed during the session period to prevent it from expiring. Each refresh operation creates a cookie valid for the default interval.

Use **aaaLogin** to establish a connection session and get a valid cookie. Use **aaaRefresh** to maintain the session and keep the cookie active. Use **aaaLogout** to terminate the session (also invalidates the cookie). A maximum of 256 sessions to the VNMC can be opened at any one time. Subsequent login requests will be rejected after the maximum session limit is reached.

Operations are done via the HTTP post method. VNMC only supports HTTPS protocol with port 443. The HTTP envelope contains the XML configuration.

Query Methods

Query methods obtain information on the current configuration state of a VNMC object. These are query examples:

- **configResolveDn**—Retrieves objects by DN
- **configResolveDns**—Retrieves objects by a set of DNs
- **configResolveClass**—Retrieves objects of a given class
- **configResolveClasses**—Retrieves objects of multiple classes
- **configFindDnsByClassId**—Retrieves the DNs of a specified class
- **configResolveChildren**—Retrieves the child objects of an object
- **configResolveParent**—Retrieves the parent object of an object
- **configScope**—Performs class queries on a DN in the management information tree

Most query methods have the argument *inHierarchical* (Boolean true/yes or false/no). If true, the *inHierarchical* argument returns all child objects. Here is an example.

```
<configResolveDn ... inHierarchical="false"></>
<configResolveDn ... inHierarchical="true"></>
```

Send document comments to vnmc-docfeedback@cisco.com.

Query API methods may also have an *inRecursive* argument to specify whether the call should be recursive (that is, follow objects that point back to other objects or the parent object).

Query Filters

The API provides a set of filters to increase the usefulness of the query methods. These filters can be passed as part of a query and are used to identify the wanted result set. Filters are categorized as:

- Simple filters
- Property filters
- Composite filters
- Modifier filters

Simple Filters

There are two simple filters, true and false. These two filters react to the simple states of true or false, respectively.

- True filter—Result set of objects with the Boolean condition of true
- False filter—Result set of objects with the Boolean condition of false

Property Filters

Property filters use the values of an object's properties as the criteria for inclusion in a result set. To create most property filters, *classId* and *propertyId* of the target object/property is required, along with a value for comparison.

- Equality filter—Restricts the result set to objects with the identified property of “equal” to the provided property value.
- Not equal filter—Restricts the result set to objects with the identified property of “not equal” to the provided property value.
- Greater than filter—Restricts the result set to objects with the identified property of “greater than” the provided property value.
- Greater than or equal filter—Restricts the result set to objects with the identified property of “is greater than or equal” to the provided property value.
- Less than filter—Restricts the result set to objects with the identified property of “less than” the provided property value.
- Less than or equal filter—Restricts the result set to objects with the identified property of “less than or equal” to the provided property value.
- Wildcard filter—Restricts the result set to objects with the identified property matches that includes a wildcard. Supported wildcards include “%” or “*” (any sequence of characters), “?” or “-” (any single character).
- Any bits filter—Restricts the result set to objects with the identified property that has at least one of the passed bits set. (Use only on bitmask properties.)
- All bits filter—Restricts the result set to objects with the identified property that has all the passed bits set. (Use only on bitmask properties.)

Send document comments to vnmc-docfeedback@cisco.com.

Composite Filters

Composite filters are composed of two or more component filters. They enable greater flexibility in creating result sets. For example, a composite filter could restrict the result set to only those objects that were accepted by at least one of the contained filters.

- AND filter—Result set must pass the filtering criteria of each component filter. For example, to obtain all compute blades with totalMemory greater than 64 megabytes and operability of operable, the filter is composed of one greater than filter and one equality filter.
- OR filter—Result set must pass the filtering criteria of at least one of the component filters. For example, to obtain all the service profiles that have an *assignmentState* of unassigned or an association state value of unassociated, the filter is composed of two equality filters.
- Between filter—Result set is those objects that fall between the range of the first specified value and second specified value. For example, all faults that occurred between two dates.
- XOR filter—Result set is those objects that pass the filtering criteria of no more than one of the composite's component filters.

Modifier Filters

Modifier filters change the results of a contained filter. Only one modifier filter is supported, the NOT filter. This filter negates the result of a contained filter. Use this filter to obtain objects that do not match contained criteria.

Configuration Methods

There are several methods to make configuration changes to managed objects. These changes can be applied to the whole tree, a subtree, or an individual object. Examples of these methods are:

- configConfMo—Affects a single subtree (that is, a DN)
- configConfMos—Affects multiple subtrees (that is, several DNs)
- configConfMoGroup—Makes the same configuration changes to multiple subtree structures (DNs) or managed objects.

Most configuration methods use the argument *inHierarchical* (Boolean true/yes or false/no). These values do not play a significant role during configuration because child objects are included in the XML document and the DME operates in the forgiving mode.

Event Subscription Methods

When an object is created, changed, or deleted because of a user- or system-initiated action, an event is generated. Applications can get VNMC state change information by regular polling or event subscription. Because polling is resource-expensive, event subscription is the preferred method of notification.

Event subscription allows a client application to register for event notification from the VNMC. When an event occurs, VNMC the subscribing client applications of the event and its type. Only actual change events are sent, not the object's unaffected attributes. This applies to all object changes in the system.

To subscribe to VNMC event notification, open an HTTP session and keep the session open. Then post the eventSubscribe request thru the HTTP session as showing in the following example:

Send document comments to vnmc-docfeedback@cisco.com.

```
<eventSubscribe cookie="<real_cookie>"></eventSubscribe>
```

After the eventSubscribe request is accepted by VNMCM, it starts sending all new events as they occur through the HTTP session. To get a valid cookie for event subscription, you must log into VNMCM first. If not logged in, the event subscription request will be rejected with an error response.

Each event has a unique event ID. These event IDs operate as counters and are included in all event notifications. When an event is generated, the event ID counter increases (a new event is assigned a new event ID). This enables tracking of events and ensures no event is missed. If an event is missed by the client (identifiable by a missing event ID), use **loggingSyncOcn**s to retrieve the missed event.

**Note**

For VNMCM 1.0, only HTTP protocol is supported for event subscription. Use **HTTP** (do not use **HTTPS**) when posting the event subscription request.

Capturing XML Interchange between the GUI and VNMCM Server

VNMCM GUI is a web-based application. Capture the XML interchange between the browser GUI client and the VNMCM server to learn the real API usage. Since VNMCM is developed using Adobe FLEX GUI framework, install a debug version of the Adobe flash player (from Adobe website). This will capture the log output stored in a log file under the user's home directory. In Windows 7, the log file can be found under C:\Users\<username>\AppData\Roaming\Macromedia\Flex Player\Logs\flashlog.txt). For example, most of the sample request and response payload specified in VNMCM XML API Use Case Example sections are captured in this manner.

Success or Failure Responses

The VNMCM responds almost immediately to any API request. If the request cannot be completed, a failure is returned. For a query or login method, the actual results are returned. For configuration methods, a successful response will indicate that the request is valid, but not indicate that the operation was completed. For example, after a DB backup request is accepted with a successful response from VNMCM, the VNMCM server may take a longer time to finish the actual backup job.

This section contains the following topics:

- [Successful Response, page 1-10](#)
- [Failed Requests, page 1-11](#)
- [Empty Results, page 1-11](#)

Successful Response

Upon success, an XML document is returned with the requested information or a confirmation that changes were made. The following example is a configResolveDn request on a policy with the DN "org-root/org-tenant d3337/pol-pl"

```
<configResolveDn cookie="<real_cookie>"
  dn="org-root/org-tenant_d3337/pol-pl"
  inHierarchical="false"/>
```

The response includes the following details

Send document comments to vnmc-docfeedback@cisco.com.

```
<configResolveDn
  dn="org-root/org-tenant_d3337/pol-p1"
  cookie="<real_cookie>"
  commCookie="7/13/0/a3c"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
<outConfig>
  <policyRuleBasedPolicy
    descr=""
    dn="org-root/org-tenant_d3337/pol-p1"
    intId="10811"
    name="p1"/>
  </outConfig>
</configResolveDn></configResolveDn>
```

Failed Requests

Response to failed request includes XML attributes for errorCode and errorDescr. The following is an example of a failed request, when trying to create a policy that already exists in the system:

```
<configConfMo
  dn="org-root/org-tenant_d3337/pol-p1"
  cookie="<real_cookie>"
  commCookie="7/13/0/2038"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes"
  errorCode="103"
  invocationResult="unidentified-fail"
  errorDescr="can't create; object already exists.">
</configConfMo>
```

Empty Results

A query request for a nonexistent object is not treated as a failure by the DME. If the object does not exist, a success message is returned, but the XML document contains an empty data field (<outConfig> </outConfig>) which indicates that the requested object was not found. The following is an example of resolve by DN on a non-existent policy.

```
<configResolveDn
  dn="org-root/org-tenant_d3337/pol-p1"
  cookie="<real_cookie>"
  commCookie="7/13/0/203e"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfig>
  </outConfig>
</configResolveDn>
```

Send document comments to vnmc-docfeedback@cisco.com.



CHAPTER 2

Using the VNMC XML API

This chapter includes the following:

- [Tutorial Examples, page 2-1](#)
- [VNMC XML API Use Case Example, page 2-12](#)

API method details are provided below.

Tutorial Examples

This section contains the following topics:

- [Authentication Methods, page 2-1](#)
- [Query Methods, page 2-3](#)
- [Using Query Methods for Policies, page 2-6](#)
- [Using Filters, page 2-7](#)

Authentication Methods

Authentication allows API interaction with VNMC. It also provides a way to set permissions and control the operations that can be performed.



Note

A session cookie is a 47-character string; it is not the type of cookie that web browsers store locally to maintain session information. Most code examples use “<real_cookie>” for an actual cookie (such as 1217377205/85f7ff49-e4ec-42fc-9437-da77a1a2c4bf).

Login

The following example uses a Linux POST command to post authentication requests, using HTTPS to VNMC.

```
POST https://10.193.34.70/xmlIM/mgmt-controller
Please enter content (application/x-www-form-urlencoded) to be POSTed:
```

Then use **aaaLogin** and provide a username and password:

```
<aaaLogin
  inName="admin"
```

Send document comments to vnmc-docfeedback@cisco.com.

```
inPassword="cisco@123" />
```

**Note**

The XML version and DOCTYPEs are not present and should not be used. The inName and inPassword attributes are parameters.

Each XMLAPI document represents an operation to be performed. When the request is received as an XML API document, VNMCM reads the request and performs the actions as provided in the method. VNMCM responds with a message in XML document format and indicates success or failure of the request.

The following is a typical successful response:

```
(1) <aaaLogin
(2)     response="yes"
(3)     outCookie="<real_cookie>"
(4)         outRefreshPeriod="600"
(5)         outPriv="admin,read-only"
(6)         outDomains="mgmt02-dummy"
(7)         outChannel="fullssl"
(8)         outEvtChannel="fullssl">
(9) </aaaLogin>
```

- (1) The method used to login.
- (2) Confirms this is a response.
- (3) This is the session cookie.
- (4) The recommended cookie refresh period. The default login session length is two hours.
- (5) The privileges assigned to the user account.
- (6) The outDomains value is mgmt02-dummy.
- (7) The outChannel value is fullssl (this session is using encryption over SSL).
- (8) The outEvtChannel value is fullssl (any event subscriptions use encryption over SSL).
- (9) Closing tag.

**Note**

VNMCM 1.0 event channel is using HTTP so it is neither encrypted nor over SSL

Refreshing the Session

Sessions are refreshed with **aaaRefresh** method, using the 47-character cookie obtained either from the **aaaLogin** response or a previous refresh.

```
<aaaRefresh
  inName="admin"
  inPassword="mypassword"
  inCookie="<real_cookie>" />
```

Logging Out of the Session

Use the following method to log out of a session:

```
<aaaLogout
  inCookie="<real_cookie>" />
```

Unsuccessful Responses

The following is the response to a failed login:

```
(1) <aaaLogin
(2)     response="yes"
```

Send document comments to vnmc-docfeedback@cisco.com.

```
(3)      cookie=" "  
(4)      errorCode="551"  
(5)      invocationResult="unidentified-fail"  
(6)      errorDescr="Authentication failed">  
(7) </aaaLogin>
```

- (1) The method used to login. The VNMC sends a response message with the result from the method.
- (2) Confirms this is a response.
- (3) The VNMC does not supply a session cookie.
- (4) (5) (6) Shows the error code and description.
- (7) Closing tag

Query Methods

Query methods obtain information (that is, hierarchy, state, and scope)

This section contains the following topics:

- [Using configResolveDn, page 2-3](#)
- [Using configResolveDns, page 2-3](#)
- [Using configResolveClass, page 2-4](#)
- [Using configResolveClasses, page 2-4](#)
- [Using configFindDnsByClassId, page 2-4](#)
- [Using configResolveChildren, page 2-5](#)
- [Using configResolveParent, page 2-5](#)
- [Using configScope, page 2-5](#)

Using configResolveDn

When resolving a DN, note the following:

- The object specified by the DN is retrieved
- Specified DN identifies the object instance to be resolved (required)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

See the example request/response in [configResolveDn, page 3-22](#).

Using configResolveDns

When resolving multiple DNs, note the following:

- The objects specified by the DNs are retrieved
- Specified DN identifies the object instance to be resolved (required)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

Send document comments to vnmc-docfeedback@cisco.com.

- Order of a request does not determine the order of the response

Unknown DNSs are returned as part of the *outUnresolved* element.

See the example request/response in [configResolveDns](#), page 3-24.

Using configResolveClass

When resolving a class, note the following:

- All objects of the specified class type are retrieved
- *classId* specifies the object class name to return (required)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

Result sets can be large. Be precise when defining result sets. For example, to obtain only a list of servers, use *computeBlade* as the attribute value for *classId* in the query. To get all instances of equipment, query the *equipmentItem* class. This queries for all instances of the class *equipmentItem*:

```
<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="faultPolicy" />
```

See the example request/response in [configResolveClass](#), page 3-20.

Using configResolveClasses

When resolving multiple classes, note the following:

- This retrieves all the objects of the specified class types
- *classId* attribute specifies the name of the object class to return (required)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

If an invalid class name is specified in the *inId* attribute, an XML parsing error is generated. The query cannot execute.

See the example request/response in [configResolveClasses](#), page 3-21.

Using configFindDnsByClassId

When finding distinguished names of a specified class, note the following

- This retrieves the DNSs of a specified class
- *classId* specifies the object type to retrieve (required)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

See the example request/response in [configFindDnsByClassId](#), page 3-16.

Send document comments to vnmc-docfeedback@cisco.com.

Using configResolveChildren

When resolving children of objects in the management information tree, note the following:

- This obtains all child objects of a named object that are instances of the named class. If a class name is omitted, all child objects of the named object are returned.
- *inDn* attribute specifies the named object from which the child objects are retrieved (required).
- *classId* attribute specifies the name of the child object class to return (optional).
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

See the example request/response in [configResolveChildren, page 3-18](#).

Using configResolveParent

When resolving the parent object of an object, note the following:

- This retrieves the parent object of a specified DN
- *dn* attribute is the DN of the child object (required)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

See the example request/response in [configResolveParent, page 3-26](#).

Using configScope

Limiting the scope of a query allows for a finer grained, less resource-intensive request. The query can be anchored at a point in the management information tree other than the root. When setting the query scope, note the following:

- This sets the root (scope) of the query to a specified DN and returns objects of the specified class type
- *dn* is the named object from which the query is scoped (required)
- *inClass* attribute specifies the name of the object class to return (optional; when a class is not specified, the query acts the same as **configResolveDn**)
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required
- *inHierarchical* attribute (default = false) if true, specifies that results are hierarchical
- Enumerated values, classIds, and bit masks are displayed as strings

The following example queries for the policies in the policy manager:

```
<configScope dn="sys/chassis-1"
  inClass=""
  cookie="<real_cookie>"
  inHierarchical="false"/>
```

Also see the example request/response in [configScope, page 3-28](#).

Send document comments to vnmc-docfeedback@cisco.com.

Using Query Methods for Policies

Policies are available on different organizations. In a large system with many policies, querying all policies at once is resource intensive. Instead, identify the type of policy and the parent org on which the policies are attached. For example, use the following query to get all the rule based policies defined under a tenant with the dn "org-root/org-tenant_d3338",

```
<configScope
  cookie="<real_cookie>"
  inHierarchical="false"
  dn="org-root/org-tenant_d3338"
  inClass="policyRuleBasedPolicy" />
```

The response is as follows:

```
<configScope
  dn="org-root/org-tenant_d3338"
  cookie="<real_cookie>"
  commCookie="7/13/0/24e7"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <policyRuleBasedPolicy
      descr=" "
      dn="org-root/org-tenant_d3338/pol-p9"
      intId="10274"
      name="p9" />
    <policyRuleBasedPolicy
      descr=" "
      dn="org-root/org-tenant_d3338/pol-p1"
      intId="10301"
      name="p1" />
  </outConfigs>
</configScope>
```

Although, a query by the DN is more efficient:

```
<configResolveDn
  inHierarchical="false"
  cookie="<real_cookie>"
  dn="sys org-root/org-tenant_d3338/pol-p1">
</configResolveDn>
```

To get the policy object and its rule data, change inHierarchical to true:

```
<configResolveDn
  inHierarchical="true"
  cookie="<real_cookie>"
  dn=" org-root/org-tenant_d3338/pol-p1">
</configResolveDn>
```

Querying Faults

The following example obtains a list of major faults:

```
<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="faultInst" />
```

Send document comments to vnmc-docfeedback@cisco.com.

The following example (which contains the filter `<inFilter>eq class="faultInst"`) obtains a list of all major or critical faults:

```
<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="faultInst">
  <inFilter>
    <eq class="faultInst"
      property="highestSeverity"
      value="major" />
  </inFilter>
</configResolveClass>
```

Using Filters

Use filters to create specific and targeted queries. This section contains the following topics:

- [Simple Filters, page 2-7](#)
- [Property Filters, page 2-7](#)
- [Composite Filters, page 2-10](#)
- [Modifier Filter, page 2-11](#)

Simple Filters

Simple filters use true and false conditions of properties to select results.

False example

```
<configResolveClass
  cookie="<real_cookie>"
  classId="topSystem"
  inHierarchical="false">
  <inFilter>
  </inFilter>
</configResolveClass>
```

True example

```
<configResolveClass
  cookie="<real_cookie>"
  classId="topSystem"
  inHierarchical="true">
  <inFilter>
  </inFilter>
</configResolveClass>
```

Property Filters

Equality Filter

The example shows a query for all associated compute firewalls. The equality filter is framed as follows:

```
<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
```

Send document comments to vnmc-docfeedback@cisco.com.

```

classId="fwComputeFirewall">
  <inFilter>
    <eq class="fwComputeFirewall"
      property="assocState"
      value="associated" />
  </inFilter>
</configResolveClass>

```

Not Equal Filter

The example finds all compute firewalls that are not associated. The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="fwComputeFirewall">
  <inFilter>
    <ne class="fwComputeFirewall"
      property="assocState"
      value="associated" />
  </inFilter>
</configResolveClass>

```

Greater Than Filter

The example finds the memory arrays with more than 1024 MB capacity. The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="memoryArray">
  <inFilter>
    <gt class="memoryArray"
      property="currCapacity"
      value="1024" />
  </inFilter>
</configResolveClass>

```

Greater Than or Equal to Filter

The example finds core dump files with size greater than 2MB. The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="sysdebugCore">
  <inFilter>
    <ge class="sysdebugCore"
      property="size"
      value="2097152" />
  </inFilter>
</configResolveClass>

```

Less Than Filter

The example finds core dumpfiles with size less than 2MB. The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="sysdebugCore">
  <inFilter>
    <lt class="sysdebugCore"

```


Send document comments to vnmc-docfeedback@cisco.com.

```

        property="size"
        value="2097152" />
    </inFilter>
</configResolveClass>

```

Less Than or Equal to Filter

The example finds core dump files with size less than or equals to 2MB. The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="sysdebugCore">
  <inFilter>
    <le class="sysdebugCore"
      property="size"
      value="2097152" />
  </inFilter>
</configResolveClass>

```

Wildcard Filter

The wildcard filter uses standard regular expression syntax. The example finds any virtual firewalls whose name begins with the prefix “dmz”:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="fwInstance">
  <inFilter>
    <wcard class="fwInstance"
      property="name"
      value="dmz.*" />
  </inFilter>
</configResolveClass>

```

Any Bits Filter

The example finds all managed endpoints that have a *capacity* of either vm-fw or vm-vasw (the property capability is a bit mask). The filter is framed as follows:

```

<configResolveClass
  cookie="null"
  inHierarchical="false"
  classId="extpolClient">
  <inFilter>
    <anybit class="extpolClient"
      property="capability"
      value="vm-fw,vm-vasw" />
  </inFilter>
</configResolveClass>

```

All Bits Filter

The example finds all service profiles with the *capability* bit mask set to both vm-fw and vm-slb. The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="extpolClient">
  <inFilter>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

    <allbits class="extpolClient"
      property="capability"
      value="vm-fw,vm-vasw" />
  </inFilter>
</configResolveClass>

```

Composite Filters

AND Filter

To find all computer firewalls that are associated with configuration properly applied, run the following query:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="fwComputeFirewall">
  <inFilter>
    <and>
      <eq class="fwComputeFirewall"
        property="assocState"
        value="associated" />
      <eq class="fwComputeFirewall"
        property="configState"
        value="applied" />
    </and>
  </inFilter>
</configResolveClass>

```

OR Filter

The example returns all managed endpoints whose operational state is either “lost-visibility” or “unregistered”.

```

<configResolveClass
  inHierarchical="false"
  cookie="<real_cookie>"
  classId="extpolClient">
  <inFilter>
    <or>
      <eq class="extpolClient"
        property="operState"
        value="unregistered"/>
      <eq class="extpolClient"
        property="operState"
        value="lost-visibility"/>
    </or>
  </inFilter>
</configResolveClass>

```

Between Filter

The example finds the memory arrays with slots 1, 2, 3, 4, or 5 populated (note that the between range is inclusive). The filter is framed as follows:

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="memoryarray">
  <inFilter>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

    <bw class="memoryArray"
        property="populated"
        firstValue="1"
        secondValue="5" />
  </inFilter>
</configResolveClass>

```

AND, OR, NOT Composite Filter

The example is an AND, OR, NOT combination. It returns all objects of type computeBlade that are located in slot one or slot eight from all chassis, except chassis five.

```

<configResolveClass
  inHierarchical="false"
  cookie="<real_cookie>"
  classId="computeBlade">
  <inFilter>
    <and>
      <or>
        <eq class="computeBlade"
            property="slotId"
            value="1" />
        <eq class="computeBlade"
            property="slotId"
            value="8" />
      </or>
      <not>
        <eq class="computeBlade"
            property="chassisId"
            value="5" />
      </not>
    </and>
  </inFilter>
</configResolveClass>

```

Modifier Filter

NOT Filter

The NOT filter can negate a contained filter. The example below queries for managed endpoints that do not have a capability of “vm-fw” (the property capability is a bit mask):

```

<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="extpolClient">
  <inFilter>
    <not>
      <anybit class="extpolClient"
          property="capability"
          value="vm-fw" />
    </not>
  </inFilter>
</configResolveClass>

```

Send document comments to vnmc-docfeedback@cisco.com.

VNMC XML API Use Case Example

This use case is provided to enable API developers to have an example on using the VNMC XML API to configure and manage the VNMC application.

This use case example is divided into these sections with the ordering of sections representing the workflow:

- [Authentication via Management Controller, page 2-12](#)
- [Tenant Management via Service Registry, page 2-13](#)
- [Policy Management, page 2-13](#)
- [Resource Management, page 2-23](#)

Authentication via Management Controller

Login

Access to the VNMC is session-based and must first be authenticated with a login request. Default session lengths are 120 minutes. Sessions can be extended with the `aaaKeepAlive` and `aaaRefresh` methods. It is recommended that when activity is completed that the user manually log out of the session using `aaaLogout`.

Use service type 'mgmt-controller' to send your login request to VNMC.

Request

POST URL: `https://10.193.33.221/xmlIM/mgmt-controller`

XML API payload:

```
<aaaLogin
  inName="admin"
  inPassword="Nbv12345" />
```

Response

```
<aaaLogin
  cookie=" "
  commCookie=" "
  srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc=" " destSvc=" "
  response="yes"
  outCookie="<real_cookie>"
  outRefreshPeriod="600"
  outPriv="admin, read-only"
  outDomains=" "
  outChannel="fullssl"
  outEvtChannel="fullssl"
  outSessionId="web_13528"
  outVersion="1.0">
</aaaLogin>
```

Send document comments to vnmc-docfeedback@cisco.com.

Tenant Management via Service Registry

The Service Registry (service type 'service-reg') is used to create and manage tenants and the sub-organizations which are used to organize the policies and resources under a tree hierarchy for the bottom-up based policy and resource resolution. Class names used in tenant and its sub-org creation VNMCM supports the following four level organization hierarchies:

- Tenant (class: orgTenant)—Top level organization to define a tenant
- DataCenter (class: orgDatacenter)—defines data centers under a tenant
- App (class: orgApp)—Defines applications under a data center
- Tier (class: orgTier)—Defines the application tiers (web, db, app) that a VM runs

To create a new or update an existing org object, provide the object's DN and attribute values and use status "created" or "modified" respectively. To delete an org object, use status="deleted" in the request. The request shows how to create a tenant named *demoTenant*.

Request

```
POST URL: https://10.193.33.221/xmlIM/service-reg
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/org-demoTenant">
      <orgTenant dn="org-root/org-demoTenant"
                name="demoTenant"
                status="created"/>
    </pair>
  </inConfigs>
</configConfMos>
```

Response

```
<configConfMos
  cookie="<real_cookie>"
  commCookie="2/15/0/839"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="service-reg_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/org-demoTenant">
      <orgTenant
        descr=""
        dn="org-root/org-demoTenant"
        fltAggr="0"
        level="1"
        name="demoTenant"
        status="created"/>
    </pair>
  </outConfigs>
</configConfMos>
```

Policy Management

The API sample payloads have been taken from their usage in the VNMCM GUI application.

In each example where **configConfMos** API used a single XML element <pair> to specify a single configuration object, as an alternative, the **configConfMo** API can be used to obtain the same result.

Send document comments to vnmc-docfeedback@cisco.com.

This section includes the following topics:

- [Device Policy, page 2-14](#)
- [Device Profile, page 2-17](#)
- [Zone, page 2-18](#)
- [Object Group, page 2-19](#)
- [Attribute Dictionary, page 2-20](#)
- [Policy, page 2-20](#)
- [PolicySet, page 2-22](#)
- [Security Profile, page 2-22](#)

Device Policy

These actions establish various policies that include:

- [Syslog, page 2-14](#)
- [SNMP, page 2-15](#)
- [LogProfile, page 2-16](#)

Syslog

Description

The syslog policy object tracks all actions that take place within the system. It sets the logging level for a device policy. This creates **mysyslog**.

Request

```
POST URL: https://10.193.33.221/xmlIM/policy-mgr
XML API payload:
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/syslog-mysyslog">
      <commSyslog dn="org-root/syslog-mysyslog">
        <commSyslogConsole
          adminState="enabled"
          severity="emergencies"/>
        <commSyslogClient
          name="primary"
          adminState="enabled"
          hostname="5.6.7.8"
          severity="notifications"
          forwardingFacility="local7"/>
        <commSyslogClient
          name="secondary"
          adminState="enabled"
          hostname="123.23.53.123"
          severity="warnings"
          forwardingFacility="local5"/>
      </commSyslog>
    </pair>
  </inConfigs>
</configConfMos>
```

Response

Send document comments to vnmc-docfeedback@cisco.com.

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1b9"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/syslog-mysyslog">
      <commSyslog
        adminState="enabled"
        descr=""
        dn="org-root/syslog-mysyslog"
        intId="25301"
        name="mysyslog"
        port="514"
        proto="udp"
        severity="warnings"
        status="created"/>
      </pair>
    </outConfigs>
  </configConfMos>

```

SNMP

This example creates a SNMP policy named **mysnmp**. Once created, this will be available by that name in the device profile listing.

Request

POST URL: <https://10.193.33.221/xmlIM/policy-mgr>

XML API payload:

```

<configConfMos
  cookie="<real_cookie>">
  <inConfigs>
    <pair key="org-root/snmp-mysnmp">
      <commSnmp dn="org-root/snmp-mysnmp"
        adminState="enabled"
        descr="The default SNMP policy"
        sysContact="Andrew Jackson"
        sysLocation="San Jose" >
      <commSnmpCommunity
        community="public"
        role="read-only"/>
      <commSnmpTrap
        hostname="nms-1.cisco.com"
        community="private"/>
      <commSnmpTrap
        hostname="nms-2.cisco.com"
        community="private"/>
      </commSnmp>
    </pair>
  </inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1bc"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"

```

Send document comments to `vnmc-docfeedback@cisco.com`.

```

destSvc="policy-mgr_dme"
response="yes">
<outConfigs>
  <pair key="org-root/snmp-mysnmp">
    <commSnmp
      adminState="enabled"
      descr="The default SNMP policy"
      dn="org-root/snmp-mysnmp"
      intId="25281"
      name="mysnmp"
      port="161"
      proto="all"
      sysContact="Andrew Jackson"
      sysLocation="San Jose"/>
    </pair>
  </outConfigs>
</configConfMos>

```

LogProfile

This example sets the debug level for a LogProfile policy named **debugLog**. It sets the logging level.

Request

POST URL: `https://10.193.33.221/xmlIM/policy-mgr`

XML API payload:

```

<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/logprof-default">
      <policyLogProfile
        dn="org-root/logprof-debugLog"
        name="debugLog"
        level="debug1"
        size="1000000"
        backupCount="4"/>
      </pair>
    </inConfigs>
  </configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/33"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/logprof-debugLog">
      <policyLogProfile
        adminState="enabled"
        backupCount="4"
        descr="the log level for every process"
        dn="org-root/logprof-debugLog"
        intId="10068"
        level="debug1"
        name="debugLog"
        size="1000000"
        status="modified"/>
      </pair>
    </outConfigs>

```


Send document comments to vnmc-docfeedback@cisco.com.

```
</configConfMos>
```

Device Profile

This action creates a device profile named **myDeviceProfile**. It enables the profile and designates the contents of the profile.

Request

POST URL: <https://10.193.33.221/xmlIM/policy-mgr>

XML API payload:

```
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/org-Cola/fwdevprofile-myDeviceProfile">
      <fwpolicyFirewallDeviceProfile
        dn="org-root/org-Cola/fwdevprofile-myDeviceProfile"
        adminState="enabled"
        snmpPolicy="mysnmp"
        syslogPolicy="mysyslog"
        timezone="America/Los_Angeles" >
      <commDns
        name="somedns.com"
        domain="somedns.com"/>
      <!-- The order means the device should first use the DNS provider with the smallest
      "order" value. If that DNS server is not responsive, use the DNS provider with the next
      smaller number. -->
      <commDnsProvider
        hostip="171.70.168.183"
        order="1"/>
      <commDnsProvider
        hostip="171.68.226.120"
        order="2"/>
      <commDnsProvider
        hostip="64.102.6.247"
        order="3"/>
      <commNtpProvider
        name="somentp.com"
        order="1"/>
      <commNtpProvider
        name="north-america.pool.ntp.org"
        order="2"/>
      </fwpolicyFirewallDeviceProfile>
    </pair>
  </inConfigs>
</configConfMos>
```

Response

```
<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1ba"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/org-Cola/fwdevprofile-myDeviceProfile">
      <fwpolicyFirewallDeviceProfile
        adminState="enabled"
        coreFilePolicy=""
        descr=""
```

Send document comments to vnmc-docfeedback@cisco.com.

```

    dn="org-root/org-Cola/fwdevprofile-myDeviceProfile"
    dnsPolicy=""
    enablePolicyDecisionLog="no"
    faultPolicy=""
    httpPolicy=""
    httpsPolicy=""
    intId="25326"
    logProfilePolicy=""
    name="myDeviceProfile"
    snmpPolicy="mysnmp"
    status="created"
    syslogPolicy="mysyslog"
    telnetPolicy=""
    timezone="America/Los_Angeles"/>
  </pair>
</outConfigs>
</configConfMos>

```

Zone

This creates two zones named **trustedClients-0** and **trustedServers-0**. A set of attributes with values are also assigned.

Request

POST URL: <https://10.193.33.221/xmlIM/policy-mgr>

XML API payload:

```

<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/org-Cisco/zone-trustedClients-0">
<!-- Create a zone of all VMs in the 192.168.1.0/24 subnet -->
      <policyZone dn="org-root/org-Cisco/zone-trustedClients-0">
        <policyZoneCondition id="1" order="1">
          <policyZoneExpression opr="prefix">
            <policyIPAddress id="1" value="192.168.1.0" />
            <policyIPSubnet id="1" value="255.255.255.0" />
          </policyZoneExpression>
        </policyZoneCondition>
      </policyZone>
    </pair>
    <pair key="org-root/org-Cisco/zone-trustedServers-0">
<!-- Create a zone of all VMs attached to a VNSP where the "appType" is set to
"BuildServer" -->
      <policyZone dn="org-root/org-Cisco/zone-trustedServers-0">
        <policyZoneCondition id="1" order="1">
          <policyZoneExpression opr="eq">
            <policyParentAppName id="1" value="BuildServer" />
          </policyZoneExpression>
        </policyZoneCondition>
      </policyZone>
    </pair>
  </inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1a6"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"

```

Send document comments to vnmc-docfeedback@cisco.com.

```

destSvc="policy-mgr_dme"
response="yes">
<outConfigs>
  <pair key="org-root/org-tenant0/zone-zone0">
    <policyZone
      descr=""
      dn="org-root/org-tenant0/zone-zone0"
      intId="24295"
      name="zone0"
      status="created"/>
  </pair>
  <pair key="org-root/org-Cisco/zone-trustedServers-0">
    <policyZone
      descr=""
      dn="org-root/org-Cisco/zone-trustedServers-0"
      intId="24404"
      name="trustedServers-0"
      status="created"/>
  </pair>
  <pair key="org-root/org-Cisco/zone-trustedClients-0">
    <policyZone
      descr=""
      dn="org-root/org-Cisco/zone-trustedClients-0"
      intId="24408"
      name="trustedClients-0"
      status="created"/>
  </pair>
</outConfigs>
</configConfMos>

```

Object Group

This example creates an object group named **ftpPorts0** and assigns a set of attributes.

Request

POST URL: <https://10.193.33.221/xmlIM/policy-mgr>

XML API payload:

```

<configConfMos
  cookie="<real_cookie>">
  <inConfigs>
    <pair key="org-root/org-tenant0/objgrp-ftpPorts0">
      <policyObjectGroup dn="org-root/org-tenant0/objgrp-ftpPorts0">
        <policyObjectGroupExpression id="1" order="1" opr="eq">
          <policyNetworkPort id="1" value="20" />
        </policyObjectGroupExpression>
        <policyObjectGroupExpression id="2" order="2" opr="eq">
          <policyNetworkPort id="1" value="21" />
        </policyObjectGroupExpression>
      </policyObjectGroup>
    </pair>
  </inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1a4"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"

```

Send document comments to vnmc-docfeedback@cisco.com.

```

response="yes">
<outConfigs>
  <pair key="org-root/org-tenant0/objgrp-ftpPorts0">
    <policyObjectGroup
      attributeName=""
      descr=""
      dn="org-root/org-tenant0/objgrp-ftpPorts0"
      intId="24265"
      name="ftpPorts0"
      status="created" />
    </pair>
  </outConfigs>
</configConfMos>

```

Attribute Dictionary

This sets the dictionary that defines the various attribute IDs and name.

Request

```

POST URL: https://10.193.33.221/xmlIM/policy-mgr
XML API payload:
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
<!-- Create Sample VnspCustomDictionary under org-Cisco -->
    <pair key="org-root/attr-dict-custom-userAttrs">
      <policyVnspCustomDictionary dn="org-root/attr-dict-custom-userAttrs">
        <policyVnspCustomAttr dataType="string" id="1" name="userAttr1" />
        <policyVnspCustomAttr dataType="string" id="2" name="userAttr2" />
        <policyVnspCustomAttr dataType="string" id="3" name="dept" />
        <policyVnspCustomAttr dataType="string" id="4" name="production" />
      </policyVnspCustomDictionary>
    </pair>
  </inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1a3"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
<outConfigs>
  <pair key="org-root/attr-dict-custom-userAttrs">
    <policyVnspCustomDictionary
      descr=""
      dn="org-root/attr-dict-custom-userAttrs"
      intId="24245"
      name="userAttrs"
      status="created" />
    </pair>
  </outConfigs>
</configConfMos>

```

Policy

This creates a policy named **trustedHosts** and sets the rules it can use.

Send document comments to vnmc-docfeedback@cisco.com.**Request**

```

POST URL: https://10.193.33.221/xmlIM/policy-mgr
XML API payload:
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/org-Cisco/pol-trustedHosts">
      <policyRuleBasedPolicy dn="org-root/org-Cisco/pol-trustedHosts">
        <policyRule name="allowSsh" order="1">
<!-- This rule allows all VMs in zone "trustedClients" to initiate an SSH connection to
VMs in zone "trustedServers" -->
          <policyRuleCondition id="100" order="1">
            <policyNetworkExpression opr="eq">
              <policyNwAttrQualifier attrEp="source"/>
              <policyZoneNameRef id="1" value="trustedClients-0" />
            </policyNetworkExpression>
          </policyRuleCondition>
          <policyRuleCondition id="101" order="20">
            <policyNetworkExpression opr="eq">
              <policyNwAttrQualifier attrEp="destination"/>
              <policyZoneNameRef id="1" value="trustedServers-0" />
            </policyNetworkExpression>
          </policyRuleCondition>
          <policyRuleCondition id="103" order="30">
            <policyNetworkExpression opr="eq">
              <policyNwAttrQualifier attrEp="destination"/>
              <policyNetworkPort id="1" placement="0" value="22" />
            </policyNetworkExpression>
          </policyRuleCondition>
          <fwpolicyAction actionType="permit"/>
        </policyRule>
        <policyRule name="allowTacacs" order="2">
          <fwpolicyAction actionType="permit"/>
        </policyRule>
      </policyRuleBasedPolicy>
    </pair>
  </inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1b5"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/org-Cisco/pol-trustedHosts">
      <policyRuleBasedPolicy
        descr=""
        dn="org-root/org-Cisco/pol-trustedHosts"
        intId="25131"
        name="trustedHosts"
        status="created"/>
    </pair>
  </outConfigs>
</configConfMos>

```

Send document comments to vnmc-docfeedback@cisco.com.**PolicySet**

This creates **myPolicySet0** and sets the order that policies are applied.

Request

POST URL: <https://10.193.33.221/xmlIM/policy-mgr>

XML API payload:

```
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/org-tenant0/pset-myPolicySet0">
      <policyPolicySet
        dn="org-root/org-tenant0/pset-myPolicySet0"
        <!-- Policy Set contains references to three policies. Policies are resolved by name in
        the org hierarchy. Ordering of policies is important, so we use the "order" property to
        order the policies. Note that two policy sets can refer to the same policies, and use a
        different order. -->
        <policyPolicyNameRef order="1" policyName="trustedHosts"/>
      </policyPolicySet>
    </pair>
  </inConfigs>
</configConfMos>
```

Response

```
<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1a8"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/org-tenant0/pset-myPolicySet0">
      <policyPolicySet
        adminState="enabled"
        descr=""
        dn="org-root/org-tenant0/pset-myPolicySet0"
        intId="24431"
        name="myPolicySet0"
        status="created"/>
    </pair>
  </outConfigs>
</configConfMos>
```

Security Profile

This creates a security profile named **vnsp-sp1**, and assigns the VNSP to it.

Request

POST URL: <https://10.193.33.221/xmlIM/policy-mgr>

XML API payload:

```
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/org-tenant0/vnsp-sp1">
      <policyVirtualNetworkServiceProfile
        dn="org-root/org-tenant0/vnsp-sp1"
        policySetNameRef="myPolicySet0">
        <policyVnspAVPair id="1">
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        <policyAttributeDesignator attrName="dept" />
        <policyAttributeValue value="DEV" />
    </policyVnspAVPair>
</policyVirtualNetworkServiceProfile>
</pair>
</inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1ac"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
<outConfigs>
  <pair key="org-root/org-tenant0/vnsp-sp1">
    <policyVirtualNetworkServiceProfile
      descr=" "
      dn="org-root/org-tenant0/vnsp-sp1"
      intId="24512"
      name="sp1"
      policySetNameRef="myPolicySet0"
      status="created"
      vnspId="2" />
  </pair>
</outConfigs>
</configConfMos>

```

Resource Management

This component performs the following functions:

- Assigns the compute firewall to a device policy.
- Queries for available firewall instances
- Associate firewall instances with a managed object

This section includes the following topics:

- [Compute Firewall, page 2-23](#)
- [Query Firewall Instances, page 2-24](#)
- [Associate Firewall Instance, page 2-25](#)

Compute Firewall

This assigns **computeFirewall** to a device policy..

Request

```

POST URL: https://10.193.33.221/xmlIM/resource-mgr
XML API payload:
<configConfMos
  cookie="<real_cookie>">
  <inConfigs>
    <pair key="org-root/org-Cisco/cfw-VSNaaa">

```

Send document comments to vnmc-docfeedback@cisco.com.

```

    <fwComputeFirewall dn="org-root/org-Cisco/cfw-VSNaaa"
      devicePolicy="myDeviceProfile"
      hostname="cfw-VSNaaa"
      dataIpAddress="10.10.10.100">
    </fwComputeFirewall>
  </pair>
</inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="5/15/0/19a"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/org-Cisco/cfw-VSNaaa">
      <fwComputeFirewall
        assocState="unassociated"
        configState="not-applied"
        dataIpAddress="10.10.10.100"
        dataIpSubnet="255.255.255.0"
        descr=" "
        devicePolicy="myDeviceProfile"
        dn="org-root/org-Cisco/cfw-VSNaaa"
        fltAggr="0"
        hostname="cfw-VSNaaa"
        intId="12368"
        name="VSNaaa"
        status="created" />
      </pair>
    </outConfigs>
  </configConfMos>

```

Query Firewall Instances

This queries all firewall instances and their attributes.

Request

POST URL: <https://10.193.33.221/xmlIM/resource-mgr>

XML API payload:

```

<configResolveClass
  cookie="<real_cookie>"
  classId="fwInstance"
  inHierarchical="false">
  <inFilter>
    <eq class="fwInstance"
      property="mgmtIp"
      value="10.193.33.221" />
  </inFilter>
</configResolveClass>

```

Response

```

configResolveClass
  cookie=<real_cookie>"
  commCookie="5/15/0/1d9"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"

```


Send document comments to vnmc-docfeedback@cisco.com.

```

srcSvc="sam_extXMLApi"
destSvc="resource-mgr_dme"
response="yes"
classId="fwInstance">
<outConfigs>
  <fwInstance
    assignedToDn=" "
    assoc="none"
    descr=" "
    dn="fw/inst-1005"
    fltAggr="0"
    fsmDescr=" "
    fsmPrev="DisassociateSuccess"
    fsmProgr="100"
    fsmRmtInvErrCode="none"
    fsmRmtInvErrDescr=" "
    fsmRmtInvRslt=" "
    fsmStageDescr=" "
    fsmStamp="2010-12-03T23:18:13.304"
    fsmStatus="nop"
    fsmTry="0"
    intId="10155"
    mgmtIp="10.193.33.221"
    model=" "
    name="firewall"
    pooled="0"
    registeredClientDn="extpol/reg/clients/client-1005"
    revision="0"
    serial=" "
    svcId="1005"
    vendor=" " />
  </outConfigs>
</configResolveClass>

```

Associate Firewall Instance

This associates a firewall instance to a managed object.

This example assigns **firewall-0** to the **inst-1005**.

Request

```

POST URL: https://10.193.33.221/xmlIM/resource-mgr
XML API payload:
<configConfMos
  cookie="<real_cookie>">
  <inConfigs>
    <pair key="org-root/org-Cola/cfw-firewall-0">
      <fwComputeFirewall dn="org-root/org-Cola/cfw-firewall-0">
        <fwResourceBinding assignedToDn="fw/inst-1005"/>
      </fwComputeFirewall>
    </pair>
  </inConfigs>
</configConfMos>

```

Response

```

<configConfMos
  cookie="<real_cookie>"
  commCookie="5/15/0/1df"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"

```

Send document comments to vnmc-docfeedback@cisco.com.

```
response="yes">
<outConfigs>
  <pair key="org-root/org-Cola/cfw-firewall-0">
    <fwComputeFirewall
      assocState="associated"
      configState="not-applied"
      dataIpAddress="168.1.1.221"
      dataIpSubnet="255.255.255.0"
      descr=""
      devicePolicy="myDeviceProfile"
      dn="org-root/org-Cola/cfw-firewall-0"
      fltAggr="1"
      hostname="cfw-firewall-0"
      intId="12514"
      name="firewall-0"
      status="modified"/>
    </pair>
  </outConfigs>
</configConfMos>
```



CHAPTER 3

VNMC XML API Method Descriptions

This chapter includes the following:

- [VNMC XML API Methods, page 3-1](#)

API method details are provided below.

VNMC XML API Methods

These methods are also called from the GUI console. This section provides API method descriptions, syntax (request and response), and a usage example. The API methods for VNMC are defined as follows.

- [aaaGetRemoteUserRoles](#)
- [aaaGetUserLocales](#)
- [aaaKeepAlive](#)
- [aaaLogin](#)
- [aaaLogout](#)
- [aaaRefresh](#)
- [configConfFiltered](#)
- [configConfMo](#)
- [configConfMoGroup](#)
- [configConfMos](#)
- [configFindDnsByClassId](#)
- [configConfMoGroup](#)
- [configMoChangeEvent](#)
- [configResolveClass](#)
- [configResolveClasses](#)
- [configResolveDn](#)
- [configResolveDns](#)
- [configResolveParent](#)
- [configScope](#)
- [eventSendHeartbeat](#)
- [eventSubscribe](#)
- [eventSubscribeApps](#)
- [faultAckFault](#)
- [faultAckFaults](#)
- [faultResolveFault](#)
- [loggingSyncOcns](#)
- [methodVessel](#)
- [orgResolveElements](#)
- [orgResolveInScope](#)
- [orgResolveLogicalParents](#)
- [policyEstimateImpact](#)
- [poolResolveInScope](#)

aaaGetRemoteUserRoles

(core - authentication) This returns user privileges for the remote location.

Request syntax

```
<xs:element name="aaaGetRemoteUserRoles" type="aaaGetRemoteUserRoles"
substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaGetRemoteUserRoles" mixed="true">
    <xs:attribute name="inRemoteUserName">
      <xs:simpleType>
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z][a-zA-Z0-9_@-]{0,31}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
</xs:complexType>

```

Response syntax

```

<xs:element name="aaaGetRemoteUserRoles" type="aaaGetRemoteUserRoles"
substitutionGroup="externalMethod"/>
    <xs:complexType name="aaaGetRemoteUserRoles" mixed="true">
        <xs:attribute name="outRemoteUserPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((policy|aaa|read-only|admin|tenant|operations|res-config|fault),){0,7}(policy|aaa|
read-only|admin|tenant|operations|res-config|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="cookie" type="xs:string"/>
        <xs:attribute name="response" type="YesOrNo"/>
        <xs:attribute name="errorCode" type="xs:unsignedInt"/>
        <xs:attribute name="errorDescr" type="xs:string"/>
        <xs:attribute name="invocationResult" type="xs:string"/>
    </xs:complexType>

```

Examples**Request**

```

<aaaGetRemoteUserRoles
  cookie="<real_cookie>"
  inRemoteUserName="adminuser"
  outRemoteUserPriv/>

```

Response

```

<aaaGetRemoteUserRoles
  cookie="<real_cookie>"
  commCookie="11/15/0/2964"
  srcExtSys="10.193.33.109"
  destExtSys="10.193.33.109"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes"
  outRemoteUserPriv="admin">
</aaaGetRemoteUserRoles>

```

aaaGetUserLocales

(core - query) This returns a list of authorized user locales.

Send document comments to vnmc-docfeedback@cisco.com.

Request syntax

```
<xs:element name="aaaGetUserLocales" type="aaaGetUserLocales"
substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaGetUserLocales" mixed="true">
    <xs:attribute name="inUserName">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="[a-zA-Z][a-zA-Z0-9_@-]{0,31}"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="inIsUserRemote">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="aaaGetUserLocales" type="aaaGetUserLocales"
substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaGetUserLocales" mixed="true">
    <xs:attribute name="outUserLocales">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="0"/>
          <xs:maxLength value="512"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples

Request

```
<aaaGetUserLocales
  cookie="<real_cookie>"
  inUserName="john"
  inIsUserRemote="no"
  outUserLocales/>
```

Response

```
<aaaGetUserLocales
```

Send document comments to vnmc-docfeedback@cisco.com.

```

cookie="<real_cookie>"
commCookie="11/15/0/2962"
srcExtSys="10.193.33.109"
destExtSys="10.193.33.109"
srcSvc="sam_extXMLApi"
destSvc="mgmt-controller_dme"
response="yes"
outUserLocales="TestSanity">
</aaaGetUserLocales>

```

aaaKeepAlive

(generic - authentication) This keeps the session active until the default session time expires. This uses the same cookie after the method call.

Request syntax

```

<xs:element name="aaaKeepAlive" type="aaaKeepAlive" substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaKeepAlive" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="aaaKeepAlive" type="aaaKeepAlive" substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaKeepAlive" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>

```

Examples

Request

```

<aaaKeepAlive
  cookie="<real_cookie>" />

```

Response

```

<aaaKeepAlive
  cookie="<real_cookie>"
  commCookie="11/15/0/2969"
  srcExtSys="10.193.33.109"
  destExtSys="10.193.33.109"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes">
</aaaKeepAlive>

```

Send document comments to vnmc-docfeedback@cisco.com.

aaaLogin

(generic - authentication) This is the login process. This is required to begin a session. This establishes an authenticated https session between the client and the VNMC.

Request syntax

```
<xs:element name="aaaLogin" type="aaaLogin" substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaLogin" mixed="true">
    <xs:attribute name="inName">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="[\-\.\:_a-zA-Z0-9]{0,16}" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="inPassword">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="0" />
          <xs:maxLength value="512" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string" />
    <xs:attribute name="response" type="YesOrNo" />
  </xs:complexType>
```

Response syntax

```
<xs:element name="aaaLogin" type="aaaLogin" substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaLogin" mixed="true">
    <xs:attribute name="outCookie" type="xs:string" />
    <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt" />
    <xs:attribute name="outPriv">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern
value="((policy|aaa|read-only|admin|tenant|operations|res-config|fault),){0,7}(policy|aaa|
read-only|admin|tenant|operations|res-config|fault){0,1}" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="outDomains" type="xs:string" />
    <xs:attribute name="outChannel">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="fullssl" />
          <xs:enumeration value="noencssl" />
          <xs:enumeration value="plain" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="outEvtChannel">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="fullssl" />
          <xs:enumeration value="noencssl" />
          <xs:enumeration value="plain" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="outSessionId">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:minLength value="1"/>
                <xs:maxLength value="64"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="outVersion" type="xs:string"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
</xs:complexType>

```

Examples**Request**

```

<aaaLogin
  inName="admin"
  inPassword="Nbv12345" />

```

Response

```

<aaaLogin cookie=" "
  commCookie=" "
  srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc=" " destSvc=" "
  response="yes"
  outCookie="<real_cookie>"
  outRefreshPeriod="600"
  outPriv="admin"
  outDomains=" "
  outChannel="fullssl"
  outEvtChannel="fullssl"
  outSessionId="web_49019"
  outVersion="1.0(0.39938) ">
</aaaLogin>

```

aaaLogout

(generic - authentication) This is the logout process to end a current session. When the default session time period expires, this is called automatically.

Request syntax

```

<xs:element name="aaaLogout" type="aaaLogout" substitutionGroup="externalMethod"/>
<xs:complexType name="aaaLogout" mixed="true">
  <xs:attribute name="inCookie" type="xs:string"/>
  <xs:attribute name="cookie" type="xs:string"/>
  <xs:attribute name="response" type="YesOrNo"/>
</xs:complexType>

```


Send document comments to vnmc-docfeedback@cisco.com.

Response syntax

```
<xs:element name="aaaLogout" type="aaaLogout" substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaLogout" mixed="true">
    <xs:attribute name="outStatus">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="success"/>
          <xs:enumeration value="failure"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples

Request

```
<aaaLogout
  inCookie="<real_cookie>"
  outStatus/>
```

Response

```
<aaaLogout cookie=" "
  commCookie=" "
  srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc=" "
  destSvc=" "
  response="yes"
  outStatus="success">
</aaaLogout>
```

aaaRefresh

(generic - authentication) Sessions can be kept active (within the default session time frame) by user activity. There is a default of 7200 seconds that counts down when inactivity begins. If the 7200 seconds expire, VNMC enters a sleep mode. It requires signing back in, which restarts the countdown. It continues using the same session ID.



Note

Using this method will expire the previous cookie and issue a new cookie.

Request syntax

```
<xs:element name="aaaRefresh" type="aaaRefresh" substitutionGroup="externalMethod"/>
  <xs:complexType name="aaaRefresh" mixed="true">
    <xs:attribute name="inName">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="[\-\.\.:_a-zA-Z0-9]{0,16}"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="inPassword">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:minLength value="0"/>
            <xs:maxLength value="512"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="inCookie" type="xs:string"/>
<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
</xs:complexType>

```

Response syntax

```

<xs:element name="aaaRefresh" type="aaaRefresh" substitutionGroup="externalMethod"/>
    <xs:complexType name="aaaRefresh" mixed="true">
        <xs:attribute name="outCookie" type="xs:string"/>
        <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
        <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="( (policy|aaa|read-only|admin|tenant|operations|res-config|fault), ) {0,7} (policy|aaa|
read-only|admin|tenant|operations|res-config|fault) {0,1}"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="outDomains" type="xs:string"/>
        <xs:attribute name="outChannel">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="fullssl"/>
                    <xs:enumeration value="noencssl"/>
                    <xs:enumeration value="plain"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="outEvtChannel">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="fullssl"/>
                    <xs:enumeration value="noencssl"/>
                    <xs:enumeration value="plain"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="cookie" type="xs:string"/>
        <xs:attribute name="response" type="YesOrNo"/>
        <xs:attribute name="errorCode" type="xs:unsignedInt"/>
        <xs:attribute name="errorDesc" type="xs:string"/>
        <xs:attribute name="invocationResult" type="xs:string"/>
    </xs:complexType>

```

Examples**Request**

Send document comments to vnmc-docfeedback@cisco.com.

```
<aaaRefresh
  cookie="<real_cookie>"
  inName="admin"
  inPassword="Nbv12345"
  inCookie="<real_cookie>"/>
```

Response

```
<aaaRefresh
  cookie="<real_cookie>"
  commCookie=" " srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc=" "
  destSvc=" "
  response="yes"
  outCookie="<real_cookie>"
  outRefreshPeriod="600"
  outPriv="admin"
  outDomains=" "
  outChannel="fullssl"
  outEvtChannel="fullssl">
</aaaRefresh>
```

configConfFiltered

(generic - configuration) Data and activity is limited according to the configured policies.

Request syntax

```
<xs:element name="configConfFiltered" type="configConfFiltered"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfFiltered" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
      <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="classId" type="namingClassId"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="configConfFiltered" type="configConfFiltered"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfFiltered" mixed="true">
```

Send document comments to vnmc-docfeedback@cisco.com.

```

<xs:all>
  <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
</xs:all>
<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
<xs:attribute name="errorCode" type="xs:unsignedInt"/>
<xs:attribute name="errorDescr" type="xs:string"/>
<xs:attribute name="invocationResult" type="xs:string"/>
<xs:attribute name="classId" type="namingClassId"/>
</xs:complexType>

```

Examples**Request**

```

<configConfFiltered
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="orgTenant">
  <inFilter>
    <eq class="orgTenant"
      property="name"
      value="Cisco" />
  </inFilter>
  <inConfig>
    <orgDatacenter
      dn="org-HR"
      descr="HR (Human Resources- new Descr)"/>
  </inConfig>
</configConfFiltered>

```

Response

```

<configConfFiltered
  cookie="<real_cookie>"
  commCookie="5/15/0/617"
  srcExtSys="10.193.33.206"
  destExtSys="10.193.33.206"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes"
  classId="orgTenant">
  <outConfigs>
    <orgDatacenter
      descr="HR (Human Resources- new Descr) "
      dn="org-root/org-Cisco/org-HR"
      fltAggr="0"
      level="2"
      name="HR"
      status="modified"/>
  </outConfigs>
</configConfFiltered>

```

configConfMo

(generic - configuration) The configures the specified Managed Object in a single subtree (i.e., DN).

Send document comments to vnmc-docfeedback@cisco.com.

Request syntax

```
<xs:element name="configConfMo" type="configConfMo" substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMo" mixed="true">
    <xs:all>
      <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
      <xs:attribute name="cookie" type="xs:string"/>
      <xs:attribute name="response" type="YesOrNo"/>
      <xs:attribute name="dn" type="referenceObject"/>
    </xs:complexType>
```

Response syntax

```
<xs:element name="configConfMo" type="configConfMo" substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMo" mixed="true">
    <xs:all>
      <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Examples

Request

```
<configConfMo
  dn=" "
  cookie="real_cookie"
  inHierarchical="false">
  <inConfig>
    <aaaLdapEp
      attribute="CiscoAvPair"
      basedn="dc=pasadena,dc=cisco,dc=com"
      descr=" "
      dn="sys/ldap-ext"
      filter="sAMAccountName=$userid"
      retries="1"
      status="modified"
      timeout="30"/>
    </inConfig>
  </configConfMo>
```

Response

Send document comments to vnmc-docfeedback@cisco.com.

```

<configConfMo
  dn=" "
  cookie="<real_cookie>"
  commCookie="11/15/0/28"
  srcExtSys="10.193.33.101"
  destExtSys="10.193.33.101"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes">
<outConfig>
  <aaaLdapEp
    attribute="CiscoAvPair"
    basedn="dc=pasadena,dc=cisco,dc=com"
    childAction="deleteNonPresent"
    descr=" "
    dn="sys/ldap-ext"
    filter="sAMAccountName=$userid"
    fsmDescr=" "
    fsmPrev="updateEpSuccess"
    fsmProgr="100"
    fsmStageDescr=" "
    fsmStamp="2010-11-22T23:41:01.826"
    fsmStatus="nop"
    fsmTry="0"
    intId="10027"
    name=" "
    retries="1"
    status="modified"
    timeout="30"/>
  </outConfig>
</configConfMo>

```

configConfMoGroup

(generic - configuration) This configures groups of managed objects based upon the configured policies.

Request syntax

```

<xs:element name="configConfMoGroup" type="configConfMoGroup"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMoGroup" mixed="true">
    <xs:all>
      <xs:element name="inDns" type="dnSet" minOccurs="0"/>
      <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Send document comments to vnmc-docfeedback@cisco.com.

```
</xs:complexType>
```

Response syntax

```
<xs:element name="configConfMoGroup" type="configConfMoGroup"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMoGroup" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples**Note**

The 'descr' property of orgDataCenter (under org-root/org-Cisco and org-root/org-Cola) is modified. Since the 'descr' property is not implicit, it can be modified. If implicit, the modification would not apply and a new orgDataCenter would be created.

Request

```
<configConfMoGroup
  cookie="<real_cookie>"
  inHierarchical="false">
  <inDns>
    <dn value="org-root/org-Cisco" />
    <dn value="org-root/org-Cola" />
  </inDns>
  <inConfig>
    <orgDatacenter
      dn="org-HR"
      descr="HR (Human Resources)"/>
    </inConfig>
  </configConfMoGroup>
```

Response

```
<configConfMoGroup
  cookie="<real_cookie>"
  commCookie="5/15/0/600"
  srcExtSys="10.193.33.206"
  destExtSys="10.193.33.206"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
  <outConfigs>
    <orgDatacenter
      descr="HR (Human Resources) "
      dn="org-root/org-Cola/org-HR"
      fltAggr="0"
      level="2"
      name="HR"
      status="modified"/>
    <orgDatacenter
      descr="HR (Human Resources) "
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        dn="org-root/org-Cisco/org-HR"
        fltAggr="0"
        level="2"
        name="HR"
        status="modified"/>
    </outConfigs>
</configConfMoGroup>

```

configConfMos

(generic - configuration) This configures managed objects in multiple subtrees using DNs.

Request syntax

```

<xs:element name="configConfMos" type="configConfMos" substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMos" mixed="true">
    <xs:all>
      <xs:element name="inConfigs" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_2">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="configConfMos" type="configConfMos" substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMos" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_4">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>

```


Send document comments to vnmc-docfeedback@cisco.com.

Examples

Request

```
<configConfMos
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/logprof-default">
      <policyLogProfile dn="org-root/logprof-default"
        name="default"
        level="debug1"
        size="10000000"
        backupCount="4"/>
    </pair>

    <!-- Update Controller Device Profile -->
    <pair key="org-root/controller-profile-default">
      <policyControllerDeviceProfile
        dn="org-root/controller-profile-default"
        adminState="enabled">

        <commDnsProvider hostip="171.70.168.183" order="1"/>
        <commDnsProvider hostip="171.68.226.120" order="2"/>
        <commDnsProvider hostip="64.102.6.247" order="3"/>
      </policyControllerDeviceProfile>
    </pair>
  </inConfigs>
</configConfMos>
```

Response

```
<configConfMos
  cookie="<real_cookie>"
  commCookie="7/15/0/1a74"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="org-root/logprof-default">
      <policyLogProfile
        adminState="enabled"
        backupCount="4"
        descr="the log level for every process"
        dn="org-root/logprof-default"
        intId="10065"
        level="debug1"
        name="default"
        size="10000000"/>
    </pair>
    <pair key="org-root/controller-profile-default">
      <policyControllerDeviceProfile
        adminState="enabled"
        coreFilePolicy=""
        descr="default profile for management server virtual machine"
        dn="org-root/controller-profile-default"
        dnsPolicy=""
        faultPolicy="default"
        httpPolicy="default"
        httpsPolicy="default"
        intId="10057"
        logProfilePolicy="default"
        name="default"
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        snmpPolicy=""
        syslogPolicy=""
        telnetPolicy=""
        timezone="" />
    </pair>
</outConfigs>
</configConfMos>

```

configFindDnsByClassId

(generic - query) This finds distinguished names and returns them sorted by class ID.

Request syntax

```

<xs:element name="configFindDnsByClassId" type="configFindDnsByClassId"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configFindDnsByClassId" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string" />
    <xs:attribute name="response" type="YesOrNo" />
    <xs:attribute name="classId" type="namingClassId" />
  </xs:complexType>

```

Response syntax

```

<xs:element name="configFindDnsByClassId" type="configFindDnsByClassId"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configFindDnsByClassId" mixed="true">
    <xs:all>
      <xs:element name="outDns" type="dnSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string" />
    <xs:attribute name="response" type="YesOrNo" />
    <xs:attribute name="errorCode" type="xs:unsignedInt" />
    <xs:attribute name="errorDescr" type="xs:string" />
    <xs:attribute name="invocationResult" type="xs:string" />
    <xs:attribute name="classId" type="namingClassId" />
  </xs:complexType>

```

Examples

Request

```

<configFindDnsByClassId
  classId="computeBlade"
  cookie="<real_cookie>" />

```

Response

```

<configFindDnsByClassId
  cookie="<real_cookie>"
  response="yes"
  classId="computeBlade">
  <outDns>
    <dn value="sys/chassis-1/blade-7"/>
    <dn value="sys/chassis-1/blade-5"/>
  </outDns>
</configFindDnsByClassId>

```

Send document comments to vnmc-docfeedback@cisco.com.

```
<dn value="sys/chassis-1/blade-3" />
<dn value="sys/chassis-1/blade-1" />
</outDns>
</configFindDnsByClassId>
```

configMoChangeEvent

(generic - event) This returns change events on the specified managed object.

Request syntax

```
<xs:element name="configMoChangeEvent" type="configMoChangeEvent"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configMoChangeEvent" mixed="true">
    <xs:all>
      <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inEid" type="xs:unsignedLong"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="configMoChangeEvent" type="configMoChangeEvent"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configMoChangeEvent" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples

Request

```
<configMoChangeEvent
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="service-reg_dme"
  destSvc="sam_extXMLApi"
  inEid="71449">
  <inConfig>
    <extpolRegistry
      dn="extpol/reg"
      lastPollTs="2010-11-12T20:33:51.071"
      status="modified"/>
    </inConfig>
  </configMoChangeEvent>
```

Response

```
Event (12:35:20:675):
<eventSendHeartbeat
```

Send document comments to vnmc-docfeedback@cisco.com.

```

    cookie="0/0/0/2a74"
    commCookie=""
    srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    srcSvc="" destSvc=""
    response="yes"
    outSystemTime="2010-11-12T20:34:19.630">
</eventSendHeartbeat>

Event (12:35:20:690):
<eventSendHeartbeat
  cookie="0/0/0/2a74"
  commCookie=""
  srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc="" destSvc=""
  response="yes"
  outSystemTime="2010-11-12T20:34:19.630">
</eventSendHeartbeat>

```

configResolveChildren

(generic - query) This method retrieves children of managed objects under a specific DN in the managed information tree. A filter can be used to reduce the number of children being returned.

Request syntax

```

<xs:element name="configResolveChildren" type="configResolveChildren"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveChildren" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inDn" type="referenceObject"/>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="classId" type="namingClassId"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="configResolveChildren" type="configResolveChildren"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveChildren" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
<xs:attribute name="errorCode" type="xs:unsignedInt"/>
<xs:attribute name="errorDescr" type="xs:string"/>
<xs:attribute name="invocationResult" type="xs:string"/>
<xs:attribute name="classId" type="namingClassId"/>
</xs:complexType>

```

Examples

Request

```

<configResolveChildren
  cookie="<real_cookie>"
  classId="aaaUser"
  inDn="sys/user-ext"
  inHierarchical="false">
  <inFilter>
  </inFilter>
</configResolveChildren>

```

Response

```

<configResolveChildren
  cookie="<real_cookie>"
  commCookie="11/15/0/2a59"
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes"
  classId="aaaUser">
  <outConfigs>
    <aaaUser descr="" dn="sys/user-ext/user-doe"
      email="" expiration="never" expires="no" firstName="John" intId="12999"
      lastName="Doe" name="doe" phone="" priv="admin,read-only" pwdSet="yes"/>
    <aaaUser descr="" dn="sys/user-ext/user-jacks" email="" expiration="never"
      expires="no" firstName="Play" intId="12734" lastName="Jacks" name="jacks"
      phone="" priv="fault,operations,policy,read-only,res-config,tenant"
      pwdSet="yes"/>
    <aaaUser descr="" dn="sys/user-ext/user-admin" email="" expiration="never"
      expires="no" firstName="" intId="10052" lastName="" name="admin" phone=""
      priv="admin,read-only" pwdSet="yes"/>
    <aaaUser descr="" dn="sys/user-ext/user-over" email="" expiration="never"
      expires="no" firstName="Roll" intId="12711" lastName="Over" name="over"
      phone="" priv="fault,operations,policy,read-only,res-config,tenant"
      pwdSet="yes"/>
    <aaaUser descr="" dn="sys/user-ext/user-fun" email="" expiration="never"
      expires="no" firstName="Have" intId="12708" lastName="Fun" name="fun" phone=""
      priv="read-only" pwdSet="yes"/>
    <aaaUser descr="testuser" dn="sys/user-ext/user-aaa" email="" expiration="never"
      expires="no" firstName="a" intId="10620" lastName="aa" name="aaa" phone=""
      priv="aaa,read-only" pwdSet="no"/>
  </outConfigs>
</configResolveChildren>

```

Send document comments to vnmc-docfeedback@cisco.com.

configResolveClass

(generic - query) Returns requested managed object in a given class. If *inHierarchical*=true, contains children.

Request syntax

```
<xs:element name="configResolveClass" type="configResolveClass"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveClass" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="classId" type="namingClassId"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="configResolveClass" type="configResolveClass"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveClass" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="classId" type="namingClassId"/>
  </xs:complexType>
```

Examples

Request

```
<configResolveClass
  cookie="<real_cookie>"
  classId="pkiEp"
  inHierarchical="false">
  <inFilter>
  </inFilter>
</configResolveClass>
```

Response

Send document comments to vnmc-docfeedback@cisco.com.

```
<configResolveClass
  cookie="<real_cookie>"
  commCookie="11/15/0/2a5b"
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes"
  classId="pkiEp">
  <outConfigs>
    <pkiEp descr=" "
      dn="sys/pki-ext"
      intId="10037"
      name=" " />
  </outConfigs>
</configResolveClass>
```

configResolveClasses

(generic - query) Returns requested managed objects in several classes. If *inHierarchical*=true, contains children.

Request syntax

```
<xs:element name="configResolveClasses" type="configResolveClasses"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveClasses" mixed="true">
    <xs:all>
      <xs:element name="inIds" type="classIdSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
      <xs:attribute name="cookie" type="xs:string"/>
      <xs:attribute name="response" type="YesOrNo"/>
    </xs:complexType>
```

Response syntax

```
<xs:element name="configResolveClasses" type="configResolveClasses"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveClasses" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
```

Send document comments to vnmc-docfeedback@cisco.com.

```
</xs:complexType>
```

Examples**Request**

```
<configResolveClasses
  cookie="<real_cookie>"
  inHierarchical="false">
  <inIds>
    <Id value="computeBlade"/>
    <Id value="equipmentChassis"/>
  </inIds>
</configResolveClasses>
```

Response

(This is an abbreviated response.)

```
<configResolveClasses
  cookie="<real_cookie>"
  response="yes">
  <outConfigs>
    <computeBlade
      adminPower="policy"
      adminState="in-service"
      dn="sys/chassis-1/blade-1"
      .
      .
    </computeBlade>
    <computeBlade
      adminPower="policy"
      adminState="in-service"
      dn="sys/chassis-1/blade-3"
      .
      .
    </computeBlade>
    <computeBlade
      adminPower="policy"
      adminState="in-service"
      dn="sys/chassis-1/blade-5"
      .
      .
    </computeBlade>
    <computeBlade
      adminPower="policy"
      adminState="in-service"
      dn="sys/chassis-1/blade-7"
      .
      .
    </computeBlade>
    <equipmentChassis
      adminState="acknowledged"
      configState="ok"
      .
      .
    </equipmentChassis>
  </outConfigs>
</configResolveClasses>
```

configResolveDn

(generic - query) For a given DN, retrieves a single managed object.

Send document comments to vnmc-docfeedback@cisco.com.

Request syntax

```
<xs:element name="configResolveDn" type="configResolveDn"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveDn" mixed="true">
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="configResolveDn" type="configResolveDn"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveDn" mixed="true">
    <xs:all>
      <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Examples

Request

```
<configResolveDn
  cookie="<real_cookie>"
  dn="vmmEp/vm-mgr-vcenter1" />
```

Response

```
<configResolveDn dn="vmmEp/vm-mgr-vcenter1"
  cookie="<real_cookie>"
  commCookie="9/15/0/1c0d"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="vm-mgr_dme"
  response="yes">
  <outConfig>
    <vmManager
      adminState="enable"
      descr=""
      dn="vmmEp/vm-mgr-vcenter1"
```

Send document comments to vnmc-docfeedback@cisco.com.

```

        fltAggr="0"
        fsmDescr="AG registration with
vCenter (FSM:sam:dme:VmManagerRegisterWithVCenter) "
        fsmPrev="RegisterWithVCenterRegistering"
        fsmProgr="13"
        fsmRmtInvErrCode="none"
        fsmRmtInvErrDescr=""
        fsmRmtInvRslt=""
        fsmStageDescr="AG registration with
vCenter (FSM-STAGE:sam:dme:VmManagerRegisterWithVCenter:Registering) "
        fsmStamp="2010-11-11T21:37:15.696"
        fsmStatus="RegisterWithVCenterRegistering"
        fsmTry="1"
        hostName="savbu-vpod-dev-31.cisco.com"
        intId="21959"
        name="vcenter1"
        operState="unknown"
        stateQual=""
        type="vmware"
        version="" />
    </outConfig>
</configResolveDns>

```

configResolveDns

(generic - query) For a list of DNSs, retrieves managed objects.

Request syntax

```

<xs:element name="configResolveDns" type="configResolveDns"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveDns" mixed="true">
    <xs:all>
      <xs:element name="inDns" type="dnSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
      <xs:attribute name="cookie" type="xs:string"/>
      <xs:attribute name="response" type="YesOrNo"/>
    </xs:complexType>

```

Response syntax

```

<xs:element name="configResolveDns" type="configResolveDns"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveDns" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
      <xs:element name="outUnresolved" type="dnSet" minOccurs="0"/>
    </xs:all>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

</xs:all>
<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
<xs:attribute name="errorCode" type="xs:unsignedInt"/>
<xs:attribute name="errorDescr" type="xs:string"/>
<xs:attribute name="invocationResult" type="xs:string"/>
</xs:complexType>

```

Examples**Request**

```

<configResolveDns
  cookie="<real_cookie>"
  inHierarchical="false">
  <inDns>
    <dn value="sys/chassis-1" />
    <dn value="sys/chassis-1/blade-1/board/cpu-1" />
    <dn value="sys/chassis-1/blade-1/board/t-stats" />
  </inDns>
</configResolveDns>

```

Response

```

<configResolveDns
  cookie="<real_cookie>"
  response="yes">
  <outConfigs>
    <processorUnit
      arch="Xeon"
      cores="4"
      dn="sys/chassis-1/blade-1/board/cpu-1"
      id="1"
      model="Intel(R) Xeon(R) CPU E5520 @ 2.27GHz"
      operState="operable"
      operability="operable"
      perf="not-supported"
      power="not-supported"
      presence="equipped"
      revision="0"
      serial=""
      socketDesignation="CPU1"
      speed="2.266000"
      stepping="5"
      thermal="ok"
      threads="8"
      vendor="Intel(R) Corporation"
      voltage="ok"/>
    <equipmentChassis
      adminState="acknowledged"
      configState="ok"
      connPath="A,B"
      connStatus="A,B"
      dn="sys/chassis-1"
      fabricEpDn="fabric/server/chassis-1"
      fltAggr="2"
      fsmDescr=""
      fsmPrev="PsuPolicyConfigSuccess"
      fsmProgr="100"
      fsmRmtInvErrCode="none"
      fsmRmtInvErrDescr=""
      fsmRmtInvRslt=""
      fsmStageDescr=""

```

Send document comments to vnmc-docfeedback@cisco.com.

```

fsmStamp="2009-09-13T21:34:37"
fsmStatus="nop"
fsmTry="0"
id="1"
lcTs="1969-12-31T16:00:00"
managingInst="A"
model="N20-C6508"
operQualifier="fabric-conn-problem"
operState="fabric-conn-problem"
operability="operable"
power="ok"
presence="unknown"
revision="0"
serial="FOX1252GG84"
thermal="ok"
vendor="Cisco Systems Inc"
versionHolder="yes"/>
</outConfigs>
<outUnresolved>
  <dn value="sys/chassis-1/blade-1/board/t-stats"/>
</outUnresolved>
</configResolveDns>

```

configResolveParent

(generic - query) For a given DN, retrieves the parent of the managed object.

Request syntax

```

<xs:element name="configResolveParent" type="configResolveParent"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveParent" mixed="true">
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="configResolveParent" type="configResolveParent"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configResolveParent" mixed="true">
    <xs:all>
      <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Send document comments to vnmc-docfeedback@cisco.com.

```
<xs:attribute name="errorCode" type="xs:unsignedInt"/>
<xs:attribute name="errorDescr" type="xs:string"/>
<xs:attribute name="invocationResult" type="xs:string"/>
<xs:attribute name="dn" type="referenceObject"/>
</xs:complexType>
```

Examples

Request

```
<configResolveParent
  cookie="<real_cookie>"
  inHierarchical="false"
  dn="sys/chassis-1/blade-1/adaptor-1">
</configResolveParent>
```

Response

```
<configResolveParent dn="sys/chassis-1/blade-1/adaptor-1"
  cookie="<real_cookie>"
  response="yes">
  <outConfig>
    <computeBlade
      adminPower="policy"
      adminState="in-service"
      assignedToDn=" "
      association="none"
      availability="available"
      availableMemory="10240"
      chassisId="1"
      checkPoint="discovered"
      connPath="A,B"
      connStatus="A,B"
      descr=" "
      diagnostics="complete"
      discovery="complete"
      dn="sys/chassis-1/blade-1"
      fltAggr="0"
      fsmDescr=" "
      fsmFlags=" "
      fsmPrev="DiscoverSuccess"
      fsmProgr="100"
      fsmRmtInvErrCode="none"
      fsmRmtInvErrDescr=" "
      fsmRmtInvRslt=" "
      fsmStageDescr=" "
      fsmStamp="2009-09-23T23:44:30"
      fsmStatus="nop"
      fsmTry="0"
      intId="768052"
      lc="discovered"
      lcTs="1969-12-31T16:00:00"
      managingInst="B"
      model="N20-B6620-1"
      name=" "
      numofAdaptors="1"
      numofCores="8"
      numofCpus="2"
      numofEthHostIfs="2"
      numofFcHostIfs="0"
      numofThreads="16"
      operPower="off"
      operQualifier=" "
```

Send document comments to vnmc-docfeedback@cisco.com.

```

operState="unassociated"
operability="operable"
originalUuid="e3516842-d0a4-11dd-baad-000bab01bfd6"
presence="equipped"
revision="0"
serial="QCI12520024"
slotId="1"
totalMemory="10240"
uuid="e3516842-d0a4-11dd-baad-000bab01bfd6"
vendor="Cisco Systems Inc"/>
</outConfig>
</configResolveParent>

```

configScope

(generic - query) Returns managed objects and details about their configuration.

Request syntax

```

<xs:element name="configScope" type="configScope" substitutionGroup="externalMethod"/>
  <xs:complexType name="configScope" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inClass" type="namingClassId"/>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
    <xs:attribute name="inRecursive">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="configScope" type="configScope" substitutionGroup="externalMethod"/>
  <xs:complexType name="configScope" mixed="true"> <xs:all>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

        <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
</xs:complexType>

```

Examples**Request**

```

<configScope
  dn="org-root"
  cookie="<real_cookie>"
  inClass="orgOrgRes"
  inHierarchical="false"
  inRecursive="false">
  <inFilter>
  </inFilter>
</configScope>

```

Response

```

<configScope dn="org-root"
  cookie="<real_cookie>"
  commCookie="2/15/0/2a53"
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="sam_extXMLApi"
  destSvc="service-reg_dme"
  response="yes">
  <outConfigs>
    <orgOrgCaps
      dn="org-root/org-caps"
      org="512"
      tenant="64"/>
    <orgOrgCounts
      dn="org-root/org-counter"
      org="36"
      tenant="7"/>
  </outConfigs>
</configScope>

```

eventSendHeartbeat

(generic - event) Sends an event the indicates the current session is still active.

Request syntax

```

<xs:element name="eventSendHeartbeat" type="eventSendHeartbeat"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSendHeartbeat" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Send document comments to vnmc-docfeedback@cisco.com.**Response syntax**

```
<xs:element name="eventSendHeartbeat" type="eventSendHeartbeat"
substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSendHeartbeat" mixed="true">
    <xs:attribute name="outSystemTime" type="dateTime"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples**Request**

When the client application subscribes to an event or events via eventSubscribeApps or eventSubscribe, the VNMC will send eventSendHeartbeat periodically (default 120 seconds).

Response

```
<eventSendHeartbeat cookie="0/0/0/2a76"
  commCookie=""
  srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc=""
  destSvc=""
  response="yes"
  outSystemTime="2010-11-12T20:38:19.630">
</eventSendHeartbeat>
```

eventSubscribe

(generic - event) Subscribe request for activity.

Request syntax

```
<xs:element name="eventSubscribe" type="eventSubscribe"
substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSubscribe" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="eventSubscribe" type="eventSubscribe"
substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSubscribe" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```


Send document comments to vnmc-docfeedback@cisco.com.

```
</xs:complexType>
```

Examples

Request

```
<eventSubscribe
  cookie="<real_cookie>"
  <inFilter>
</inFilter>
</eventSubscribe>
```

Response

NO RESPONSE OR ACKNOWLEDGEMENT.

eventSubscribeApps

(generic - event) Subscribe request for activity on specified applications. For the client application to subscribe to VNMC system to receive the events from different application. In eventApplication, "ip" is the ip address for the VM where the application (DME) is running. The client application can subscribe to receive events from VSG as well, where *ip* should be the IP address for VSG, and type is "managed-endpoint".

Request syntax

```
<xs:element name="eventSubscribeApps" type="eventSubscribeApps"
substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSubscribeApps" mixed="true">
    <xs:all>
      <xs:element name="inAppList" type="configSet" minOccurs="0"/>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="eventSubscribeApps" type="eventSubscribeApps"
substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSubscribeApps" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples

Request

```
<eventSubscribeApps
  cookie="<real_cookie>"
  commCookie=" "
```

Send document comments to vnmc-docfeedback@cisco.com.

```

srcExtSys="0.0.0.0"
destExtSys="0.0.0.0"
srcSvc=""
destSvc="">
<inAppList>
  <eventApplication
    ip="10.193.33.101"
    type="service-reg"/>
  <eventApplication
    ip="10.193.33.101"
    type="policy-mgr"/>
  <eventApplication
    ip="10.193.33.101"
    type="mgmt-controller"/>
</inAppList>
<inFilter>
</inFilter>
</eventSubscribeApps>

```

Response

IF SUCCESSFUL, NO RESPONSE OR ACKNOWLEDGEMENT.

faultAckFault

(generic - event) Sends an acknowledgement when a fault is recorded.

Request syntax

```

<xs:element name="faultAckFault" type="faultAckFault" substitutionGroup="externalMethod"/>
  <xs:complexType name="faultAckFault" mixed="true">
    <xs:attribute name="inId" type="xs:unsignedLong"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="faultAckFault" type="faultAckFault" substitutionGroup="externalMethod"/>
  <xs:complexType name="faultAckFault" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>

```

Examples**Request**

```

<faultAckFault
  inHierarchical="false"
  cookie="<real_cookie>"
  inId="10120" />

```

Response

Send document comments to vnmc-docfeedback@cisco.com.

```
<faultAckFault
  cookie="<real_cookie>"
  commCookie="5/15/0/6c"
  srcExtSys="10.193.33.214"
  destExtSys="10.193.33.214"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
</faultAckFault>
```

faultAckFaults

(generic - event) Sends an acknowledgement when multiple faults are recorded.

Request syntax

```
<xs:element name="faultAckFaults" type="faultAckFaults"
substitutionGroup="externalMethod"/>
  <xs:complexType name="faultAckFaults" mixed="true">
    <xs:all>
      <xs:element name="inIds" type="idSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="faultAckFaults" type="faultAckFaults"
substitutionGroup="externalMethod"/>
  <xs:complexType name="faultAckFaults" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples

Request

```
<faultAckFaults
  cookie="<real_cookie>"
  <inIds>
    <id value="10656"/>
    <id value="10660"/>
  </inIds>
</faultAckFaults>
```

Response

```
<faultAckFaults
  cookie="<real_cookie>"
  commCookie="11/15/0/505"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
```

Send document comments to vnmc-docfeedback@cisco.com.

```

    destSvc="mgmt-controller_dme"
    response="yes">
</faultAckFaults>

```

faultResolveFault

(generic - event) Sends a response when a fault has been resolved.

Request syntax

```

<xs:element name="faultResolveFault" type="faultResolveFault"
substitutionGroup="externalMethod"/>
  <xs:complexType name="faultResolveFault" mixed="true">
    <xs:attribute name="inId" type="xs:unsignedLong"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="faultResolveFault" type="faultResolveFault"
substitutionGroup="externalMethod"/>
  <xs:complexType name="faultResolveFault" mixed="true">
    <xs:all>
      <xs:element name="outFault" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>

```

Examples

Request

```

<faultResolveFault
  inHierarchical="false"
  cookie="<real_cookie>"
  inId="10120" />

```

Response

```

<faultResolveFault
  cookie="<real_cookie>"
  commCookie="5/15/0/6a"
  srcExtSys="10.193.33.214"
  destExtSys="10.193.33.214"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
  <outFault>
    <faultInst
      ack="yes"
      cause="empty-pool"
      changeSet=" "
      code="F0135"
    >
  </outFault>
</faultResolveFault>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

        created="2010-11-19T11:02:41.568"
        descr="Virtual Security Gateway pool default is empty"
        dn="org-root/fwpool-default/fault-F0135"
        highestSeverity="minor"
        id="10120"
        lastTransition="2010-11-19T11:02:41.568"
        lc=""
        occur="1"
        origSeverity="minor"
        prevSeverity="minor"
        rule="fw-pool-empty"
        severity="minor"
        tags=""
        type="equipment"/>
    </outFault>
</faultResolveFault>

```

loggingSyncOcns

(generic - query) This retrieves event IDs from DME.

Request syntax

```

<xs:element name="loggingSyncOcns" type="loggingSyncOcns"
substitutionGroup="externalMethod"/>
  <xs:complexType name="loggingSyncOcns" mixed="true">
    <xs:attribute name="inFromOrZero" type="xs:unsignedLong"/>
    <xs:attribute name="inToOrZero" type="xs:unsignedLong"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

Response syntax

```

<xs:element name="loggingSyncOcns" type="loggingSyncOcns"
substitutionGroup="externalMethod"/>
  <xs:complexType name="loggingSyncOcns" mixed="true">
    <xs:all>
      <xs:element name="outStimuli" type="MethodSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>

```

Examples

Request

```

<loggingSyncOcns
  cookie="<real_cookie>"
  inFromOrZero="0"
  inToOrZero="4567000"/>

```

Response

Send document comments to vnmc-docfeedback@cisco.com.

List of event IDs.

methodVessel

(generic - event) This is a batch event notification that contains multiple configMoChangeEvent in a single payload.

Request syntax

```
<xs:element name="methodVessel" type="methodVessel" substitutionGroup="externalMethod"/>
  <xs:complexType name="methodVessel" mixed="true">
    <xs:all>
      <xs:element name="inStimuli" type="MethodSet" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="methodVessel" type="methodVessel" substitutionGroup="externalMethod"/>
  <xs:complexType name="methodVessel" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples**Request**

```
<methodVessel
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.77.66"
  destExtSys="10.193.77.66"
  srcSvc="resource-mgr_dme"
  destSvc="sam_extXMLApi">
  <inStimuli>
    <configMoChangeEvent
      cookie=" "
      commCookie=" "
      srcExtSys="10.193.77.66"
      destExtSys="10.193.77.66"
      srcSvc="resource-mgr_dme"
      destSvc="sam_extXMLApi"
      inEid="434920">
      <inConfig>
        <resstateNormalizedHealthState
          dn="fw/inst-1010/normalized-health-state"
          operState="config"
          status="modified"/>
      </inConfig>
    </configMoChangeEvent>
```

Send document comments to vnmc-docfeedback@cisco.com.

```

<configMoChangeEvent
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.77.66"
  destExtSys="10.193.77.66"
  srcSvc="resource-mgr_dme"
  destSvc="sam_extXMLApi"
  inEid="434921">
  <inConfig>
    <fwComputeFirewall
      configState="applying"
      dn="org-root/org-tenant_d3337/org-dc1/cfw-vsn-d3340"
      status="modified"/>
    </inConfig>
  </configMoChangeEvent>

<configMoChangeEvent
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.77.66"
  destExtSys="10.193.77.66"
  srcSvc="resource-mgr_dme"
  destSvc="sam_extXMLApi"
  inEid="434922">
  <inConfig>
    <faultInst
      dn="org-root/org-tenant_d3337/org-dc1/cfw-vsn-d3340/fault-F0117"
      lc="flapping,soaking-clear"
      status="modified"/>
    </inConfig>
  </configMoChangeEvent>
</inStimuli>
</methodVessel>
Event (10:32:2:807):

<methodVessel
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.77.66"
  destExtSys="10.193.77.66"
  srcSvc="resource-mgr_dme"
  destSvc="sam_extXMLApi">
  <inStimuli>

  <configMoChangeEvent
    cookie=" "
    commCookie=" "
    srcExtSys="10.193.77.66"
    destExtSys="10.193.77.66"
    srcSvc="resource-mgr_dme"
    destSvc="sam_extXMLApi"
    inEid="434923">
    <inConfig>
      <eventLog
        dn="event-log"
        size="1412"
        status="modified"/>
      </inConfig>
    </configMoChangeEvent>

  <configMoChangeEvent
    cookie=" "
    commCookie=" "
    srcExtSys="10.193.77.66"

```

Send document comments to vnmc-docfeedback@cisco.com.

```

destExtSys="10.193.77.66"
srcSvc="resource-mgr_dme"
destSvc="sam_extXMLApi"
inEid="434924">
<inConfig>
  <eventRecord
    affected="fw/inst-1009"
    cause="transition"
    changeSet=" "
    code="E4194509"
    created="2010-11-19T18:32:02.622"
    descr="[FSM:STAGE:REMOTE-ERROR]: WRONG STATE:Result: not-applicable Code:
unspecified Message: (sam:dme:FwInstanceAssociate:configPA)"
    dn="event-log/83081"
    id="83081"
    ind="state-transition"
    severity="info"
    status="created"
    trig="special"
    txId="219376"
    user="internal"/>
  </inConfig>
</configMoChangeEvent>
</inStimuli>
</methodVessel>

```

Response

```

<methodVessel
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.77.66"
  destExtSys="10.193.77.66"
  srcSvc="resource-mgr_dme"
  destSvc="sam_extXMLApi">
  <inStimuli>
    <configMoChangeEvent
      cookie=" "
      commCookie=" "
      srcExtSys="10.193.77.66"
      destExtSys="10.193.77.66"
      srcSvc="resource-mgr_dme"
      destSvc="sam_extXMLApi"
      inEid="434925">
      <inConfig>
        <resstateNormalizedHealthState
          dn="fw/inst-1010/normalized-health-state"
          operState="config-failure"
          status="modified"/>
      </inConfig>
    </configMoChangeEvent>

  <configMoChangeEvent
    cookie=" "
    commCookie=" "
    srcExtSys="10.193.77.66"
    destExtSys="10.193.77.66"
    srcSvc="resource-mgr_dme"
    destSvc="sam_extXMLApi"
    inEid="434926">
    <inConfig>
      <fwComputeFirewall
        configState="failed-to-apply"
        dn="org-root/org-tenant_d3337/org-dc1/cfw-vsn-d3340"

```


Send document comments to vnmc-docfeedback@cisco.com.

```

        status="modified"/>
    </inConfig>
</configMoChangeEvent>

<configMoChangeEvent
  cookie=" "
  commCookie=" "
  srcExtSys="10.193.77.66"
  destExtSys="10.193.77.66"
  srcSvc="resource-mgr_dme"
  destSvc="sam_extXMLApi"
  inEid="434927">
  <inConfig>
    <faultInst
      dn="org-root/org-tenant_d3337/org-dc1/cfw-vsn-d3340/fault-F0117"
      lastTransition="2010-11-19T18:32:02.680"
      lc="flapping"
      status="modified"/>
    </inConfig>
  </configMoChangeEvent>
</inStimuli>
</methodVessel>

```

orgResolveElements

(generic - query) Within a specified DN, retrieves managed objects that satisfy a query filter. Searches managed objects starting at an organization, and optionally in the child organizations.

If there is no organization with that DN, an empty map is returned. If found, searches managed objects with specified class and filters.

If *inHierarchical* = true, all matching objects and descendants are returned. If false, only returns matching objects. If *inSingleLevel* = true, search stops at the organization level. If false, includes child organizations.

Request syntax

```

<xs:element name="orgResolveElements" type="orgResolveElements"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveElements" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inClass" type="namingClassId"/>
    <xs:attribute name="inSingleLevel">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>

```

Send document comments to vnmc-docfeedback@cisco.com.

```

        <xs:restriction base="xs:string">
            <xs:enumeration value="no"/>
            <xs:enumeration value="yes"/>
        </xs:restriction>
    </xs:simpleType>
</xs:union>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
<xs:attribute name="dn" type="referenceObject"/>
</xs:complexType>

```

Response syntax

```

<xs:element name="orgResolveElements" type="orgResolveElements"
substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveElements" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_5">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>

```

Examples**Request**

```

<orgResolveElements
  dn="org-root/org-Cola"
  cookie="<real_cookie>"
  commCookie="7/15/0/19"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  inClass="policyPolicySet"
  inSingleLevel="no"
  inHierarchical="no">
  <inFilter>
  </inFilter>
</orgResolveElements>

```

Response

```

<orgResolveElements
  dn="org-root/org-Cola"
  cookie="<real_cookie>"
  commCookie="7/15/0/19"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"

```

Send document comments to vnmc-docfeedback@cisco.com.

```

destSvc="policy-mgr_dme"
response="yes"
errorCode="0"
errorDescr=" ">
<outConfigs>
  <pair key="pset-default">
    <policyPolicySet
      adminState="enabled"
      descr="The default Policy Set"
      dn="org-root/pset-default"
      intId="10082"
      name="default" />
  </pair>
  <pair key="pset-myPolicySet3">
    <policyPolicySet
      adminState="enabled"
      descr=" "
      dn="org-root/org-Cola/pset-myPolicySet3"
      intId="12289"
      name="myPolicySet3" />
  </pair>
  <pair key="pset-policySetSanity">
    <policyPolicySet
      adminState="enabled"
      descr=" "
      dn="org-root/org-Cola/pset-policySetSanity"
      intId="24627"
      name="policySetSanity" />
  </pair>
  <pair key="pset-pci_compliance_f">
    <policyPolicySet
      adminState="enabled"
      descr=" "
      dn="org-root/pset-pci_compliance_f"
      intId="24539"
      name="pci_compliance_f" />
  </pair>
  <pair key="pset-pci_compliance_h">
    <policyPolicySet
      adminState="enabled"
      descr=" "
      dn="org-root/pset-pci_compliance_h"
      intId="24541"
      name="pci_compliance_h" />
  </pair>
</outConfigs>
</orgResolveElements>

```

orgResolveInScope

(generic - query) With the given DN, system looks up the organization, and (optional) parent organizations recursively to the root. If no organization, an empty map is returned. If found, searches all pools with specified class and filters.



Note

If *inSingleLevel* = false, searches parent organizations up to the root directory.

Send document comments to vnmc-docfeedback@cisco.com.

Request syntax

```
<xs:element name="orgResolveInScope" type="orgResolveInScope"
substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveInScope" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inClass" type="namingClassId"/>
    <xs:attribute name="inSingleLevel">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="orgResolveInScope" type="orgResolveInScope"
substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveInScope" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_6">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Send document comments to vnmc-docfeedback@cisco.com.

Examples

Request

```
<orgResolveInScope
  cookie="<real_cookie>"
  dn="org-root/org-Cola"
  inClass="policyVirtualNetworkServiceProfile"
  inHierarchical="true"
  InSingleLevel="false" >
  <inFilter>
    <eq class="policyVirtualNetworkServiceProfile"
      property="name"
      value="spsanity" />
  </inFilter>
</orgResolveInScope>
```

Response

```
<orgResolveInScope
  dn="org-root/org-Cola"
  cookie="<real_cookie>"
  commCookie="7/15/0/1c35"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="vnsp-spsanity">
      <policyVirtualNetworkServiceProfile
        childAction="deleteNonPresent"
        descr=""
        dn="org-root/org-Cola/vnsp-spsanity"
        intId="82018"
        name="spsanity"
        policySetNameRef="policySetSanity"
        vnspId="41">
      <policyVnspAVPair
        childAction="deleteNonPresent"
        descr=""
        id="1"
        intId="82019"
        name=""
        rn="vnsp-avp-1">
        <policyAttributeValue
          childAction="deleteNonPresent"
          id="1"
          rn="attr-vall"
          value="DEV" />
        <policyAttributeDesignator
          attrName="dept"
          childAction="deleteNonPresent"
          rn="attr-ref" />
      </policyVnspAVPair>
    </policyVirtualNetworkServiceProfile>
  </pair>
  </outConfigs>
</orgResolveInScope>
```

Send document comments to vnmc-docfeedback@cisco.com.

orgResolveLogicalParents

(generic - query) With the given DN, system looks up logical parents up to the root directory.

Request syntax

```
<xs:element name="orgResolveLogicalParents" type="orgResolveLogicalParents"
substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveLogicalParents" mixed="true">
    <xs:attribute name="inSingleLevel">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Response syntax

```
<xs:element name="orgResolveLogicalParents" type="orgResolveLogicalParents"
substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveLogicalParents" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_7">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Send document comments to vnmc-docfeedback@cisco.com.

Examples

Request

```
<orgResolveLogicalParents
  dn="org-root/org-Cisco/org-HR/zone-clients"
  cookie="<real_cookie>"
  commCookie="7/15/0/12"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy_mgr_dme"
  inSingleLevel="no"
  inHierarchical="no">
</orgResolveLogicalParents>
```

Response

```
<orgResolveLogicalParents
  dn="org-root/org-Cisco/org-HR/zone-clients"
  cookie="<real_cookie>"
  commCookie="7/15/0/12"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy_mgr_dme"
  response="yes"
  errorCode="0"
  errorDescr="">
<outConfigs>
  <pair key="org-root/org-Cisco/zone-clients">
    <policyZone
      descr=""
      dn="org-root/org-Cisco/zone-clients"
      intId="24841"
      name="clients"/>
  </pair>
</outConfigs>
</orgResolveLogicalParents>
```

policyEstimateImpact

(core - query) This takes the DN and class ID of a policy managed object (MO) and returns a list of MOs impacted by this change. Use it to estimate any impacts of a policy change (created, modified, or deleted). The following are several examples:

- Modifying a name reference (that is, the resolved name will change)
- On a deletion, another object reference could refer to the nonexistent object
- On creation, an object reference can refer to a newly created object

Request syntax

```
<xs:element name="policyEstimateImpact" type="policyEstimateImpact"
substitutionGroup="externalMethod"/>
  <xs:complexType name="policyEstimateImpact" mixed="true">
    <xs:all>
      <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
```

Send document comments to vnmc-docfeedback@cisco.com.

```
<xs:attribute name="response" type="YesOrNo" />
</xs:complexType>
```

Response syntax

```
<xs:element name="policyEstimateImpact" type="policyEstimateImpact"
substitutionGroup="externalMethod"/>
  <xs:complexType name="policyEstimateImpact" mixed="true">
    <xs:all>
      <xs:element name="outImpactedMoSet" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_8">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
  </xs:complexType>
```

Examples**Request**

```
<policyEstimateImpact
  cookie="<real_cookie>"
  commCookie="7/15/0/449"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme">
  <inConfig>
    <policyRuleBasedPolicy
      dn="org-root/org-Cisco/pol-policySanity"
      status="deleted,modified"/>
    </inConfig>
  </policyEstimateImpact>
```

Response

```
<policyEstimateImpact
  cookie="<real_cookie>"
  commCookie="7/15/0/449"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes"
  errorCode="0"
  errorDescr="">
  <outImpactedMoSet>
    <pair key="org-root/org-Cisco/vnsp-spsanity">
      <policyVirtualNetworkServiceProfile
        descr=""
        dn="org-root/org-Cisco/vnsp-spsanity"
        intId="24846"
        name="spsanity"
        policySetNameRef="policySetSanity"
        vnspId="4"/>
```


Send document comments to vnmc-docfeedback@cisco.com.

```

</pair>
<pair key="org-root/org-Cisco/pset-policySetSanity/polref-policySanity">
  <policyPolicyNameRef
    dn="org-root/org-Cisco/pset-policySetSanity/polref-policySanity"
    order="1"
    policyName="policySanity"/>
</pair>
</outImpactedMoSet>
</policyEstimateImpact>

```

poolResolveInScope

(generic - query) With the given DN, system looks up the pool, and (optional) parent pools recursively to the root. If no pool exists, an empty map is returned. If found, searches all pools with specified class and filters.



Note

If *inSingleLevel* = false, searches parent pools up to the root directory.

Request syntax

```

<xs:element name="poolResolveInScope" type="poolResolveInScope"
substitutionGroup="externalMethod"/>
  <xs:complexType name="poolResolveInScope" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="inClass" type="namingClassId"/>
    <xs:attribute name="inSingleLevel">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="inHierarchical">
      <xs:simpleType>
        <xs:union memberTypes="xs:boolean">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>

```

Send document comments to vnmc-docfeedback@cisco.com.

Response syntax

```
<xs:element name="poolResolveInScope" type="poolResolveInScope"
substitutionGroup="externalMethod"/>
  <xs:complexType name="poolResolveInScope" mixed="true">
    <xs:all>
      <xs:element name="outConfigs" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_9">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
    </xs:all>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>
    <xs:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

Examples

Request

```
<poolResolveInScope
  dn="org-root/org-Cisco"
  cookie="<real_cookie>"
  class=fwPool />
```

Response

```
<poolResolveInScope
  dn="org-root/org-Cisco"
  cookie="<real_cookie>"
  commCookie="5/15/0/5bf"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
  <outConfigs>
    <pair key="fwpool-default">
      <fwPool
        assigned="0"
        descr="Default Pool of Virtual Security Gateway resources"
        dn="org-root/fwpool-default"
        fltAggr="65536"
        id="1"
        intId="10065"
        name="default"
        size="0"/>
      </pair>
    </outConfigs>
  </poolResolveInScope>
```