



CHAPTER 1

Introduction to the Cisco VNMC XML API

This chapter provides general information about the Cisco VNMC XML Application Programming Interface (API).

This chapter includes the following sections:

- [Information About Cisco VNMC and the XML API, page 1-1](#)
- [API Method Categories, page 1-7](#)
- [Capturing XML Interchange Between the Cisco VNMC GUI and the Cisco VNMC Server, page 1-10](#)
- [Success or Failure Responses, page 1-11](#)

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](#)
- [Cisco VNMC Components, page 1-2](#)
- [Overview of the Cisco VNMC XML API, page 1-5](#)
- [Cisco VNMC Data Model Schema, page 1-5](#)
- [Accessing Cisco 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, the Cisco 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, the Cisco VNMC allows you to manage Cisco 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.

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

Cisco VNMC Components

The Cisco 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.

The following sections describe these components:

- [Management Controller, page 1-3](#)
- [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 management 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 interservice API communications or the publish or subscribe method.

Send document comments to vnmc-docfeedback@cisco.com

Management Controller

The Management Controller, also known as core service, provides the system-related service for the Cisco VNMC virtual machine. It provides these services:

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

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

Service Registry

The Cisco 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, Cisco 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. It provides these services:

- Creates, deletes, and updates organizations (tenants, data centers, vApps, tiers). Organization changes are automatically propagated to the Policy Manager and Resource Manager where policies and resources are attached to the intended organizations.
- Maintains information about the registered services (providers, endpoints, management controller, service registry)
- Maintains diagnostic information, such as audit logs, faults, and 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 Cisco VSGs. When a compute firewall is associated with a Cisco VSG, the device configuration profile information (defined by the compute firewall) is pushed to the actual Cisco VSG. This triggers the Cisco VSG to download the security profiles and policies from the Policy Manager. It provides these services:

- Maintains inventory of Cisco Virtual Security Gateways (VSG) and Virtual Supervisor Modules (VSM).
- Defines compute firewalls and associates them with Cisco VSGs for provisioning.
- Integrates with VMWare vCenter instances to retrieve VM attributes.
- Maintains inventory of discovered virtual machines and distributes them to Cisco VSG instances with the following information:
 - VM attributes (name, hypervisor, parent vApp, cluster)
 - vNIC attributes (port profile name, IP addresses)
- Manages pools of Cisco VSGs.

Send document comments to vnmc-docfeedback@cisco.com

- Maintains health states and faults for VSGs.
- Maintains diagnostic information, such as audit logs, faults, and event logs.

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

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 Cisco VSG from the Resource Manager, the Cisco VSG queries the Policy Manager to resolve the device profile, security profiles, and all referenced policies. The Cisco 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 like protocol, source or destination, or IP address and port
 - Virtual machine attributes like instance name, guest OS, zone, parent app, port profile, cluster, resource pool, hostname, and 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 the Cisco VNMC system management device profiles and policies for NTP, DNS, syslog, and faults.
- Distributes policies, security profiles, device profiles, and associated objects to Cisco VSG instances.
- Maintains diagnostic information like audit logs, faults, and 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.

Send document comments to vnmc-docfeedback@cisco.com

Overview of the Cisco VNMC XML API

The Cisco VNMC XML API is a programmatic way to integrate or interact with the Cisco Virtual Network Management Center. The API interface accepts XML documents by using 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 Cisco VNMC structure with a single API call.

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 multiple managed objects (that is, policies) are being configured, 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. The Cisco 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.

The Cisco 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. The Cisco VNMC is responsible for all endpoint (Cisco 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 Cisco 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 Cisco VNMC service component's DME and is placed in the transactor queue in FIFO order. 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.

Cisco VNMC Data Model Schema

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

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

Send document comments to vnmc-docfeedback@cisco.com

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

Accessing Cisco VNMC Services

You can access Cisco VNMC services using the XML API request over HTTPS protocol. The HTTPS request URL format is:

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

<service type> is the service type of the intended service provider as described in the Cisco VNMC Components section. For example, to submit a request to the Policy Manager, use service type policy-mgr in the following 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 1-7](#)

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}...
```

Following is an example:

```
org-root/org-Cisco/zone-trustedServers-0
```

In the 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" />
```

Send document comments to vnmc-docfeedback@cisco.com

Relative Name

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

```
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, and its DN is org-root/org-Cisco/zone-trustedServers-0.

API Method Categories

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

- [Authentication Methods, page 1-7](#)
- [Query Methods, page 1-8](#)
- [Configuration Methods, page 1-9](#)
- [Event Subscription Methods, page 1-10](#)



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 Cisco 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. Following are the authentication methods:

- **aaaLogin**—This method is the initial method for logging in to the Cisco VNMC.
- **aaaRefresh**—This method refreshes the current authentication cookie.
- **aaaLogout**—This method exits the current session and deactivates the current authentication cookie.

Authentication methods initiate and maintain an active Cisco 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 Cisco VNMC can be opened at any one time. Subsequent login requests will be rejected after the maximum session limit is reached.

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

Send document comments to vnmc-docfeedback@cisco.com

Query Methods

Query methods obtain information on the current configuration state of a Cisco 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. Following is an example.

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

API Query methods also might 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`—The resulting set of objects carry the Boolean condition of `true`.
- `False filter`—The resulting set of objects carry 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, the `classId` and `propertyId` of the target object and property are 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.

Send document comments to vnmc-docfeedback@cisco.com

- 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.)

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 associationState 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 the methods include the following:

- configConfMo—Affects a single subtree, that is, a DN.
- configConfMos—Affects multiple subtrees, that is, several DNs.

Send document comments to vnmc-docfeedback@cisco.com

- `configConfMoGroup`—Makes the same configuration changes to multiple subtree structures or managed objects.

Most configuration methods use the argument `inHierarchical`. 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 the Cisco 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 Cisco VNMC. When an event occurs, the Cisco VNMC sends the subscribing client applications of the event and its type. Only actual change events are sent, not the object's unaffected attributes. This process applies to all object changes in the system.

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

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

After the `eventSubscribe` request is accepted by the Cisco VNMC, it starts sending all new events as they occur through the HTTP session. To get a valid cookie for event subscription, you must log in to the Cisco VNMC first. If not logged in, the event subscription request is 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 increments and a new event is assigned a new event ID. This process enables tracking of events and ensures that no event is missed. If an event is missed by the client, use `loggingSyncOcs` to retrieve the missed event.



Note

For the Cisco VNMC 1.0, only the HTTP protocol is supported for event subscription. Use HTTP when posting the event subscription request.

Capturing XML Interchange Between the Cisco VNMC GUI and the Cisco VNMC Server

The Cisco VNMC GUI is a web-based application. Capture the XML interchange between the GUI and the Cisco VNMC server to learn the real API usage. Because the Cisco VNMC is developed using Adobe FLEX GUI framework, install a debug version of the Adobe flash player. Installation captures 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\Flash Player\Logs\flashlog.txt`. For example, most of the sample request and response payload specified in the Cisco VNMC XML API Use Case Example sections are captured in this manner.

Send document comments to vnmc-docfeedback@cisco.com

Success or Failure Responses

The Cisco VNMC 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 the Cisco VNMC, the Cisco VNMC server might take a longer time to finish the actual backup job.

This section contains the following topics:

- [Successful Response, page 1-11](#)
- [Failed Requests, page 1-11](#)
- [Empty Results, page 1-12](#)

Successful Response

Upon a successful response, an XML document is returned with the requested information or a confirmation that changes were made.

This example shows 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:

```
<configResolveDn
  dn="org-root/org-tenant_d3337/pol-pl"
  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-pl"
      intId="10811"
      name="pl"/>
    </outConfig>
  </configResolveDn></configResolveDn>
```

Failed Requests

Response to failed requests includes XML attributes for errorCode and errorDescr. This example shows a failed request when trying to create a policy that already exists in the system:

```
<configConfMo
  dn="org-root/org-tenant_d3337/pol-pl"
  cookie="<real_cookie>"
  commCookie="7/13/0/2038"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
```

Send document comments to vnmc-docfeedback@cisco.com

```

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.

This example shows resolution by DN on a nonexistent 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>

```