Cisco Network Services Manager 5.0.2

API Version 1.1 Specification and Reference Guide

December 7, 2012

Table of Contents

1 Network Services Manager API Introduction

The Cisco Network Services Manager (Network Services Manager) 5.0.2 Northbound API (NBI) is intended to facilitate using Network Services Manager for automated network provisioning as part of a larger cloud environment. As such, the API is intended for integration with an orchestration layer, rather than for direct end-user use. Therefore, the API has limited support for granular authorization. Generally a single user/password (HTTPS/basic auth) is used for all operations, and the northbound system invoking the API is trusted.

Provisions have been made for API versioning and verifying compatibility with a particular Network Services Manager server. The first step in interacting with the Network Services Manager API should always be to confirm that it is a compatible version, as described in the section Versioning and Client Requirements.

1.1 REST overview

The Network Services Manager API attempts to follow the REST architectural style as much as possible. As originally defined[1], RESTful interfaces must adhere to the following principles:

- **Client-Server:** Separation of user-interface concerns from the server implementation, with a consistent interface between them.
- **Stateless:** No session state is maintained by the server. Each client request is processed independently, and cannot rely on session-specific context on the server. This means that all information necessary for a given request must be contained in the request.

  NOTE

  There should be no client/session state in the API, but this does not mean that there is no state in the system. The server can, of course, have state, and expose that state via REST resources. For instance, in the Network Services Manager API, all asynchronous operations create server-side "Task" objects as a side effect, which are exposed to the client and expose the server's progress in handling that asynchronous task. This is completely valid in RESTful APIs.

- **Cacheable:** The data in each response must be labeled explicitly as cacheable or non-cacheable. Performance is improved by caching when possible, but this can be done only under control of the server, using standard HTTP cache control headers.
- **Layered System:** Because of the stateless and cacheable nature of REST interactions, it is possible for a client to interact with a server through intermediary layers (caches, load balancers, security proxies, etc.) without being affected.

  Most critically, a RESTful system provides a **Uniform Interface**, according to four interface architectural constraints:

- **Identification of Resources:** The key concept of REST is the Resource. A resource is an abstract notion, and can be thought of as an addressable set of information, identified using a URI. A resource is not necessarily a single entity in the back-end system. Multiple resource URIs can pull information from the same back-end object or set of objects, depending on the model that the API is presenting. For instance, Fielding[2] gives the example of a version-controlled source file, which might represent the same underlying file data as several

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[2] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_2_1_1

resources, for "latest revision", "revision number 1.2.7", or "revision included with the Orange release".

- **Manipulation of Resources Through Representations:** Each resource is exposed through one or more representations. A representation is in a particular encoding (such as XML or JSON), and contains some or all of the data in the resource. The data in the resource is manipulated through these representations, using HTTP DELETE, PUT or POST operations.
- **Self-descriptive Messages:** Each message contains enough metadata, through cache control and media type headers, to be parsed appropriately by the client or server.
- **Hypermedia as the Engine of Application State:** Conceptually, it is possible to navigate through the resources by following links in the representations, and a client should not assign any significance to the URIs themselves. That is, a client should never construct a URI on its own, but instead traverse the existing resources, starting from a single base URI. (See http://blogs.sun.com/craigmcc/entry/why_hateoas for a more detailed justification).

  Cisco considers this to be a best practice, and has designed the Network Services Manager API to facilitate it. We also recognize that in reality, our users will write the clients for this API, and we cannot force them to not build or store URLs. Therefore, we have made some assertions about URLs and how they are allowed to change in future releases. (This is described in more detail below.)

1.2 Resources and Entities in the NBI

The Network Services Manager NBI follows these REST conventions, and therefore defines a set of *resources.* Each resource is an entity that is exposed to northbound systems, and identified with one or more URLs.

Each resource can have one or more *representation*. API users interact with the resources through these representations, which are identified through a URL and using HTTP content negotiation to specify a format (such as XML or JSON).

Content negotiation is based around *media types*, which describe the *format* of the messages being exchanged. NBI 1.1 supports only one set of representations, in XML, with a media type of `application/xml`.

We have defined several high level resource types in the NBI, as described in the following table. Each resource in the NBI will be one of the following:

ApiIndex	Located above the actual API entry point, this provides a high level overview of the API versions that are supported by a given server. For backwards compatibility, a given Network Services Manager server might support multiple versions of the NBI. If it does, each will have its own TopIndex, linked from the ApiIndex.
TopIndex	The topindex is returned when the user does a GET upon the API's main entry point. It returns a read-only index page that identifies the top-level resources supported by the NBI.
Entity	Most resources in the Network Services Manager API represent a class of objects called *entities*. Entities include resources such as tenants, service offerings, networks, and tasks. When an entity can be modified, it is done as an asynchronous operation (described below).

EntityCollection	A collection resource represents a set of entities in a deterministic order. The entitycollection itself is read-only, but it consists of entities that provide individual self links, through which they can be individually manipulated. Current entitycollections dump the entire set of entities, in-line, and thus can be quite large. Future versions of the API might support more customization of the information returned when viewing a collection.
Action	Provides a way to do anything that is not a simple update or delete operation. This includes creating new resources as well as bulk or server-assisted operations. Action resources support only the POST operation, and return a task. The data that is entered is always of a different representation than that which is returned. For instance, a create action takes in a service offering representation and returns a representation of a task resource (and eventually its result, such as a tenant or tnc).

In the XML representation of an entity, all types of entities are derived from the 'EntityValue' complex type. The information contained in an entity includes:

- **UID:** A unique resource identifier. This identifier is unique across all NBI resources on a given Network Services Manager instance, but not necessarily globally across multiple Network Services Manager instances. UIDs must be treated by clients as an opaque string, with no specific format or length. Network Services Manager 5.0.2 uses a URL syntax for NBI resource UIDs, but this is not guaranteed and should not be assumed.
- **Name:** A human-readable name for the resource. Uniqueness should not be assumed. Names must be 3 to 250 characters, begin with letter or number, and can include the characters: 0-9, A-Z, a-z, - (hyphen), _ (underscore), and space. Certain resources might provide more specific naming constraints
- **Description:** Description of the resource.
- **Version:** A version number for the resource. Whenever the resource is modified, the version number is incremented. This value must be passed back to the server as part of the PUT operation to modify, and is used to avoid collisions between multiple clients manipulating the same resource at the same time (even through multiple representations). If the resource number submitted does not match that on the server (indicating that the record has already been modified by someone else), an HTTP 409 "conflict" error is returned.
- **Properties**: A set of properties exposed by the resource. For resources that correspond to business objects managed by metamodels, the set of properties comes from the parameters and exports defined in the metamodel. Each property is defined as read-write or read-only to indicate whether or not it can be modified via the API. A number of specific data types are provided for properties, and are described below.
- **Parameters**: Only applicable to ServiceOfferings, parameters are similar to properties, but instead of describing the object, they describe a set of required inputs used when instantiating a serviceoffering. These parameters direct the creation of the entity, and become its properties.

  NOTE
  Although all parameters used to create an object will appear in its properties, not all properties come from parameters.

- **Links:** A set of links to other resources. Links are read-only, and can be omitted from the resource when performing an update operation.

1.3 Representation Formats

This version of the Network Services Manager API supports XML representations only. Future versions might include JSON support.

To indicate their desire to use XML representations, clients **must** include `Accept: application/xml` and `Content-Type: application/xml` headers for all requests. Examples of the XML format used by the API are included below, and the formal schema is made available in XSD format. (See below for details on how to retrieve this schema file.)

1.4 Parameter and Property and Data Types

It is often necessary to pass a set of parameters into an API call. For instance, when creating a network, it might be necessary to assign it an address range. This is done by passing a list of parameters, each with a type and value, represented as XML. The value is wrapped in an element that identifies its type, as shown in the following table of supported data types.

Value Type	Example XML	Value XML Type

boolean	`<property>`
	`  <name>param1</name>`
	`  <boolean>true</boolean>`
	`</property>`
		boolean

date	`<property>`
	`  <name>param1</name>`
	`  <date>2011-03-04T12:34:45-08:00</date>`
	`</property>`
		date

integer	`<property>`
	`  <name>param1</name>`
	`  <integer>123</integer>`
	`</property>`
		integer

ipv4	`<property>`
	`  <name>param1</name>`
	`  <ipv4>10.1.2.3</ipv4>`
	`</property>`
		IPv4Value

ipv4Addresses	`<property>`
	`  <name>param1</name>`
	`  <ipv4Addresses>`
	`  <ipv4>10.1.2.3</ipv4>`
	`  <ipv4>10.1.2.4</ipv4>`
	`  </ipv4Addresses>`
	`</property>`
		IPv4ListValue

ipv4Type	`<property>`
	`  <name>param1</name>`
	`  <ipv4Type>`
	`  <name>public</name>`
	`  <ipv4>0.0.0.0</ipv4>`
	`  </ipv4Type>`
	`</property>`
		IPv4TypeValue

ipv4Pair	<property> <name>param1</name> <ipv4Pair> <ipv4Type> <name>public</name> <ipv4>0.0.0.0</ipv4> </ipv4Type> <ipv4Type> <name>private</name> <ipv4>1.1.1.1</ipv4> </ipv4Type> </ipv4Pair> </property>	IPv4PairValue
ipv4Pairs	<property> <name>param1</name> <ipv4Pairs> <ipv4Pair> <ipv4Type> <name>public</name> <ipv4>0.0.0.0</ipv4> </ipv4Type> <ipv4Type> <name>private</name> <ipv4>1.1.1.1</ipv4> </ipv4Type> </ipv4Pair> <ipv4Pair> <ipv4Type> <name>three</name> <ipv4>10.0.0.3</ipv4> </ipv4Type> <ipv4Type> <name>four</name> <ipv4>10.0.0.4</ipv4> </ipv4Type> </ipv4Pair> </ipv4Pairs> </property>	IPv4PairListValue
macAddress	<property> <name>param1</name> <macAddress>01:23:45:67:89:1a</macAddress> </property>	MACAddressValue

password	<property>
	  <name>param1</name>
	  <password>somePassword</password>
	  <!-- Note: Passwords are write-only.
	  You cannot read back a password
	  after it is stored in NSM. -->
	</property>
		PasswordValue

protocolService	<property> <name>param1</name> <protocolService> <protocol>tcp</protocol> <portRange> <start>80</start> <end>97</end> </portRange> </protocolService> </property>	ProtocolServiceValue

protocolServices	<property> <name>param1</name> <protocolServices> <protocolService> <protocol>tcp</protocol> <portRange> <start>80</start> <end>97</end> </portRange> </protocolService> <protocolService> <protocol>tcp</protocol> <portRange> <start>98</start> </portRange> <redirectPort>8080</redirectPort> </protocolService> </protocolServices> </property>	ProtocolServiceListValue
qosPolicy	<property> <name>param1</name> <qosPolicy> <inPolicy>InPolicyName</inPolicy> <outPolicy>OutPolicyName</outPolicy> </qosPolicy> </property>	QosPolicyValue
qosPolicies	<property> <name>param1</name> <qosPolicies> <qosPolicy> <inPolicy>InPolicyName</inPolicy> <outPolicy>OutPolicyName</outPolicy> </qosPolicy> </qosPolicies> </property>	QosPolicyListValue
range	<property> <name>param1</name> <range> <start>123</start> <end>345</end> </range> </property>	RangeValue
ranges	<property> <name>param1</name> <ranges> <range> <start>123</start> <end>135</end> </range> <range> <start>324</start> <end>345</end> </range> </ranges> </property>	RangeListValue

routeTarget	<property> <name>param1</name> <routeTarget> <ipv4>10.40.0.0</ipv4> <ASNumber>6500</ASNumber> </routeTarget> </property> or <property> <name>param1</name> <routeTarget> <ASNumber1>6500</ASNumber1> <ASNumber2>6501</ASNumber2> </routeTarget> </property>	RouteTargetValue
routingStrategy	<property> <name>param1</name> <routingStrategy> <ospf> <area>1234</area> <devices> <routingDevice> <ipv4>10.86.241.19</ipv4> <process>4</process> </routingDevice> <routingDevice> <ipv4>10.86.241.20</ipv4> <process>4</process> </routingDevice> </devices> </ospf> </routingStrategy> </property>	RoutingStrategyValue
scheduledDate	<property> <name>param1</name> <scheduledDate> <!-- choices are IMMEDIATE or NEVER --> <schedule>NEVER</schedule> </scheduledDate> </property>	ScheduledDateValue
segmentLocation	<property> <name>param1</name> <segmentLocation> <vSphere> <vc> <ipv4>10.10.0.1</ipv4> <uuid>c4718563-b3ba-41a1-957f- 45790997f826</uuid> </vc> <dvs> <ipv4>10.11.0.3</ipv4> <uuid>91342b50-6c84-c72b-4045- e55ca3c41206</uuid> </dvs> <portGroup> <name>vlan100</name> </portGroup> </vSphere> </segmentLocation> </property>	SegmentLocationValue

segmentLocations	<property> <name>param1</name> <segmentLocations> <segmentLocation> <vSphere> <vc> <ipv4>10.10.0.1</ipv4> <uuid>c4718563-b3ba-41a1-957f-45790997f826</uuid> </vc> <dvs> <ipv4>10.11.0.3</ipv4> <uuid>91342b50-6c84-c72b-4045-e55ca3c41206</uuid> </dvs> <portGroup> <name>vlan100</name> </portGroup> </vSphere> </segmentLocation> <segmentLocation> ... </segmentLocation> </segmentLocations> </property>	SegmentLocationListValue
string	<property> <name>param1</name> <string>abc 123</string> </property>	string
strings	<property> <name>param1</name> <strings> <string>abc 123</string> <string>def 678</string> </strings> </property>	StringListValue
subnet	<property> <name>param1</name> <subnet> <ipv4>10.0.0.0</ipv4> <mask>8</mask> </subnet> </property>	SubnetValue
subnets	<property> <name>param1</name> <subnets> <subnet> <ipv4>10.0.0.0</ipv4> <mask>8</mask> </subnet> <subnet> <ipv4>192.168.1.0</ipv4> <mask>24</mask> </subnet> </subnets> </property>	SubnetListValue

trafficFilter	<!-- Coarse-grained --> <property> <name>param1</name> <trafficFilter> <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid> </trafficFilter> </property> or <!-- Fine-grained --> <property> <name>param1</name> <trafficFilter> <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid> <ipv4AddressRanges> <ipv4AddressRange> <ipv4Start>172.16.1.0</ipv4Start> <ipv4End>172.16.2.255</ipv4End> </ipv4AddressRange> </ipv4AddressRanges> </trafficFilter> </property>	TrafficFilterValue
trafficFilters	<property> <name>param1</name> <trafficFilters> <trafficFilter> <!-- as above --> </trafficFilter> <trafficFilter> <!-- as above --> </trafficFilter> </trafficFilters> </property>	TrafficFilterListValue
uid	<property> <name>param1</name> <uid>{API-URL}/v1/tenant/*tenant-001*</uid> </property>	uid
uids	<property> <name>param1</name> <uids> <uid>{API-URL}/v1/tenant/tenant-001</uid> <uid>{API-URL}/v1/tenant/tenant-002</uid> </uids> </property>	UidListValue
vrf	<property> <name>param1</name> <vrf> <name>param1</name> <devices> <vrfDevice> <ipv4>10.86.241.19</ipv4> </vrfDevice> <vrfDevice> <ipv4>10.86.241.20</ipv4> </vrfDevice> </devices> </vrf> </property>	VRFValue

When these data types are used for parameters, they are wrapped in `<parameter>` instead of `<property>`, and a `<type>` is added to indicate to the client the value type that is expected when populating the parameter value. Optionally, a default value can be included for each parameter.

Although the default for a `string` is free text entry, a mechanism is provided for a parameter to be exposed in a service offering as a selection of possible values instead. When using this mechanism, the offering contains a parameter block in the following format:

```
<parameter>
  <name>param1</name>
  <type>string</type>
  <choices>
    <choice displayValue="Choice 1" internalValue="choice1" default="true">
    <choice displayValue="Choice 2" internalValue="choice2">
  </choices>
</parameter>
```

In this case, the user must select a single choice. The response POSTed to the server must encode the `internalValue` of the selected choice it as if it were a normal `string` value:

```
<parameter>
  <name>param1</name>
  <string>choice1</string>
</parameter>
```

It is also possible to present a set of choices when more than one can be selected by using `<type>strings</type>`.

When these data types are used for presenting properties of existing entities, they will be shown as `<property>` elements. Some properties can be directly modified (via a HTTP PUT), and some cannot. The read-only properties will be indicated with the attribute `isReadOnly="true"`.

NOTE

When performing an update via PUT on a resource with read-only parameters (`isReadOnly="true"`), the read-only parameters must be included, unmodified. They cannot be omitted.

1.5 Linked Resources

Relationships between NBI resources are represented in XML as a series of `<link>` elements. The structure of an NBI `<link>` is based upon RFC4287 `<atom:link>` elements. The following attributes are provided:

href	required	This attribute will be included on every `<link>` element. The value is a URL pointing to the target resource of the relationship.
rel	required	The 'rel' attribute of a link describes the nature of the relationship to the target. The valid 'rel' values are listed in the next section.
title	optional	This provides a human-readable description of the link (generally the name of its target)
type	optional	When omitted, the target of the link should be expected to use the same media type (representation format) as the current entity. When specified, it advises the likely media type of the target of the link. This is most commonly used with `rel="alternate"` to point to alternate representations of the current resource. The type attribute is considered advisory and does not override the actual media type returned with the representation.

1.6 Supported Link Relation Types

The following types can appear as 'rel' values on `<link>` elements, following the conventions of RFC5988, Web Linking.

Rel Type	Description
self	Provides a link to the resource that contains this `<link>`.
alternate	Designates substitute versions for the representation in which the link occurs. If the alternate keyword is used with the type attribute, it indicates that the referenced document is a reformulation of the current document in the specified format.
describedby	The target resource provides a description or documentation of the current resource. The type attribute provides a media type for the target document.
child	The target resource is considered a *child* of the current resource. This means that the child cannot exist without its parent, and any delete operation performed on this resource will cascade to its children. A resource can have any number of children.
parent	The target resource is considered the *parent* of the current resource. There can be only one parent.
nsmr:catalog	The target resource represents a catalog of service offerings available for instantiation as children of the current resource.
nsmr:create	The target action resource will accept a service offering and create it, returning the created object.
nsmr:externalnetwork	The target externalnetwork resource is associated with the current externalnetworkconnection resource.
nsmr:ipaddresspool	The target ipaddresspool resource is defined with the current (network segment or container) resource.
nsmr:ipallocated	The target is a resource describing the IP allocations within the current address pool.
nsmr:ipavailable	The target is a resource describing the available IPs within the current address pool.
nsmr:ipreservations	The target is a resource describing the IP reservations within the current address pool.
nsmr:pods	The target is a collection of PODs accessible to the current resource.
nsmr:providers	The target is a collection of providers known to the system.
nsmr:superpool	The target is the ipaddresspool from which the current ipaddresspool was subdivided.
nsmr:schema	The target is a schema defining the current resource.
nsmr:tenants	The target is a collection of tenants associated with the source.

NOTE

The prefix `nsmr:` is used on all relationship types specific to Network Services Manager. This is an abbreviation of the full namespace, `http://www.cisco.com/NetworkServicesManager/rel`, using CURIE syntax. The specific prefix (nsmr) must always be used.

In documents produced by Network Services Manager, the nsmr prefix/namespace will be explicitly defined using xmlns attributes, such as the following:

```
 <links
xmlns:nsmr="http://www.cisco.com/NetworkServicesManager/rel">
   <link title="Coke" rel="parent" href="…/v1/tenant/12345"/>
   <link title="Pool" rel="nsmr:ipaddresspool"
href="…/v1/ipaddresspool/1245"/>
 </links>
```

1.7 Resource CRUD Semantics

The Network Services Manager REST API uses HTTP GET, PUT, POST, and DELETE actions, with standard HTTP behavior:

- **GET:** Retrieve an identified resource/representation. This is a read-only operation, and cannot modify any server-side state or have any side-effects.
- **POST:** Non-idempotent modification of server state. That is, a POST cannot necessarily be repeated without getting a different result than if it were sent once. In the Network Services Manager API, POST is generally used when creating new objects. The client issues a POST to a "create" URI, with a description of what is desired, and the object is created and returned, with its (new) URI. If you issue the same POST again, another object is created.
- **PUT:** Idempotent modification of server state. A representation retrieved from a resource URI using a GET request can be stored on the server using a PUT. The data passed in replaces that which was previously there. HTTP specifies that PUT is idempotent: multiple PUTs of the same data to the same resource have the same effect as one. This is not completely true of this API. Each resource's representation contains a version number, which is used to protect against multiple clients of the API modifying the same object and writing over other clients' changes. Therefore, if you PUT the same data (with the same version number) more than once, a "`409 Conflict`" response is returned. When an object is updated with a PUT, the `<links>` section, if present, is ignored.
- **DELETE:** Deletes a resource. If you delete a resource that has already been deleted, a "404 Not Found" response is returned.

If you come from a traditional object programming background, you might be more accustomed to CRUD (Create, Read, Update, Delete) semantics. These do not map perfectly to REST, and attempts to do so can lead to confusion[3]. Nonetheless, at least with this version of the Network Services Manager API, they map fairly well, and can be thought of as:

- Create = POST
- Read = GET
- Update = PUT
- Delete = DELETE

[3] http://jcalcote.wordpress.com/2008/10/16/put-or-post-the-rest-of-the-story/

Be aware that this mapping is not precise, and could change in future releases. We include this information to help illustrate how these operations are used.

1.8 Response Status Codes and Error Handling

Errors are reported according to HTTP standards, with status codes from 200-209 indicating success, and codes above 400 indicating failure. Specific codes are consistent with those described in RFC 2616. The most common codes used by this API are:

Status Code	Applicable To	Reason Phrase	Description
200	GET, POST, PUT, DELETE	OK	Request has succeeded
201	POST	Created	Async Task completed, object created
202	POST, PUT, DELETE	Accepted	Async task accepted
400	PUT, POST	Bad Request	Invalid request submitted
404	GET, POST, PUT, DELETE	Not Found	Resource Not Found
409	PUT	Conflict	Attempt to modify a resource that has already been modified by another request
500	GET, POST, PUT, DELETE	Internal Server Error	Internal Error

In the case of a success, the response body is a representation of a resource. In failures, a fault message similar to the following is returned:

```
HTTP/1.1 404 Not Found
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<fault>
  <message>Entity with Id "some-invalid-id" not found.</message>
  <details>Entity</details>
  <details>some-invalid-id</details>
</fault>
```

1.9 Asynchronous Operations

In the Network Services Manager API, all modification operations (POST, PUT, and DELETE) are asynchronous. When these operations are invoked, the server accepts the request and starts an asynchronous process to finish the operation. A tracking mechanism for long-running asynchronous operations is provided.

When such asynchronous operations are invoked, a response with the HTTP code “202 Accepted” is returned to indicate that the request has been accepted by the server for asynchronous processing. A task resource is created, and its URI is returned in the Location header of the 202 response.

The client can then issue repeated GET requests on the Task resource to monitor the progress of the asynchronous task. The Task resource is designed to capture status, operation start time, end

time, last modification time, expiry time, percent complete, errors, logs, and the object impacted due to API execution.

The following is an example of a Task corresponding to creating a tenant:

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Tenant</name>
  <links>
    <link title="Create Tenant" rel="self"
          href="{API-URL}/v1/task/task-001"/>
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-09-30T15:54:16-04:00</startTime>
  <expiryTime>2011-09-30T16:54:16-04:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

where `taskStatus` is one of the following:

`pending`	The system is currently processing the task.
`success`	The task completed as intended with the desired result. percentComplete will be 100. A <result> element will show the object that was created or modified by this task.
`failure`	A problem occurred while processing the task. percentComplete will be 100. A <fault> element will show the error details.

After the task completes, the task object resembles the following:

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Tenant Create Tenant model</name>
  <links>
    <link title="Create Tenant Create Tenant model" rel="self"
          href="{API-URL}/v1/task/task-001"/>
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-09-30T15:54:16-04:00</startTime>
  <endTime>2011-09-30T15:54:17-04:00</endTime>
  <expiryTime>2011-09-30T16:54:16-04:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="TenantValue">
    <uid>{API-URL}/v1/tenant/tenant-001</uid>
    <name>Tenant</name>
    <description>Basic Tenant</description>
      <version>2</version>
      <links>
        <link rel="self"
              href="{API-URL}/v1/tenant/tenant-001"/>
        <link rel="nsmr:catalog"
              href="{API-URL}/v1/tenant/tenant-001/catalog"/>
```

```
        <link rel="nsmr:pods"
              href="{API-URL}/v1/tenant/tenant-001/pod"/>
      </links>
      <properties/>
  </result>
</task>
```

This result indicates a successful operation. If the task had failed it would be reflected in the status, and `<fault>` would be returned in place of the `<result>` block.

Completed tasks are retained until the Network Services Manager server is restarted or the preconfigured expiration time for tasks is reached.

1.10 Versioning and Client Requirements

An NBI client's first step should always be to retrieve the API Index URL:

`http://<nsm hostname>:8443/NetworkServicesManagerAPI/`

NOTE

The version `(/v1)`, which is used in most NBI URLs, is not included. This top-level URL will tell the client which versions are supported, and provide links to their base URLs.

A typical response is:

```
<apiIndex>
  <productInformation>
       [describes installed Network Services Manager software]
  </productInformation>
  <apiVersions>
    <apiVersion>
      [describes a version of the API that is supported:
             overall media type (XML, for instance)
             schema information
             namespace information]
    </apiVersion>
  </apiVersions>
</apiIndex>
```

Product Information

`<productInformation>` contains the following elements:

`<name>`	Product Name (Network Services Manager)
`<vendor>`	Vendor (Cisco Systems)
`<version>`	Product Version
`<build>`	Build Number
`<release>`	Packaged Release Number
`<buildDate>`	Build Date
`<url>`	Product URL
`<description>`	Product Description

API Version

The client must ensure that the Network Services Manager server supports the API version that it has been written against and with which it is compatible.

For example:

```
<apiVersion>
  <version>1.1.0</version>
  <status>current</status>
  <schemaVersions>
    <schemaVersion>
      <version>1.1.0</version>
      <mediaType>application/xml</mediaType>
      <representations>
        <representation>
          <name>DEFAULT</name>
          <mediaType>application/xml</mediaType>

<namespace>http://www.cisco.com/NetworkServicesManager/1.1</namespace>

<schemaLocation>https://nsmhost:8443/NetworkServicesManagerAPI/schema/Networ
kServicesManager-1.1.0.xsd</schemaLocation>
        </representation>
      </representations>
    </schemaVersion>
  </schemaVersions>
  <links>
    <link type="application/xml" rel="self"
href="https://nsmhost:8443/NetworkServicesManagerAPI/v1"/>
    <link type="application/pdf" rel="describedby"
href="https://hsmhost:8443/NetworkServicesManagerAPI/doc/CiscoNetworkService
sManagerAPI-Guide-1.1.pdf"/>
  </links>
</apiVersion>
```

This describes a currently supported API version, 1.1.0, with an XML representation and namespace that is used for all of its resources. In addition, it provides a link to this API document.

A server can respond with multiple `<apiVersion>` values. The "status" can be one of the following:

`current`	Most recent version supported by this server
`legacy`	Indicates that the API version is legacy but still supported by the system, given the correct namespace and URLs are being used
`unsupported`	Indicates the API version is not supported

Of particular note is the "self" link of each <apiVersion>, which provides the URL of the API base URL for this API version on the server. Because the server can support multiple versions of the API, there can be multiple base URLs; as a result, it is important to consult this link to determine which URL to use. In the above example, API version 1.1.0 is accessed via https://nsmhost:8443/NetworkServicesManagerAPI/v1.

1.10.1 API versions and compatibility

Any NBI client would have been written targeting a particular API version, presumably the version that was on the server it was tested against. If the "current" version of a Network Services Manager server it is used against matches exactly, the client is obviously compatible.

An API version number uses the format *major.minor.revision*, such as 1.1.0. A client can be written so that it is compatible with any server where the revision is greater than or equal and the major and minor versions are the same as the one the client was written against.

The major version defines a set of resources and their meanings. The minor version represents a change that can be considered an extension, but is still significant enough to require the client to make small changes to ensure compatibility. Routine extensions (new resources, relationship types, etc.) are tracked via API version revisions, and should not affect client compatibility, as long as the server revision is greater than or equal to the version that the client was coded against, and the client is written correctly. In particular, the client must ignore certain things added by the server (in a newer API revision) that it does not recognize, specifically:

- New resources
- New relationships
- New XML elements as described below

Existing URLs will not stop working during a minor API version number change. Resources will not be removed until the next major version change. If this is not possible, the server may return a 410 "Gone" response.

When the server version is greater than the client's expected version, the following guidelines can be applied when determining the likely impact on a client:

API revision	No client changes should be needed, if the client was written correctly.
API minor version	The client will need testing and some small changes (such as XML namespaces and support for new resources).
API major version	The client will likely need significant changes. Changes to the API major version indicate particularly significant changes to existing resources and their meanings, and are therefore intended to be rare. All URLs will change, so any stored URLs will need to be updated. If the old major version is still supported (legacy) by the server, the URLs will be honored when possible, with `<link rel="alternate">` pointing to the new resource URL for that entity.

1.10.2 XML and Forward Compatibility

Schema Version

Each `<apiVersion>` in the API Index has one or more `<schemaVersion>` definitions, each of which applies to a specific media type. In Network Services Manager 5.0.2, only XML is supported for the NBI, so there is a single `<schemaVersion>` with a media type of `application/xml`. Each has a version number (major.minor.revision).

Each major version of the XML schema will have a distinct XML schema namespace. For instance, schemas 1.1.x will use the following namespace:

`http://www.cisco.com/NetworkServicesManager/1.1`

Therefore, each time the major or minor version number changes, the XML namespace changes. Changes to the XML schema major/minor version numbers correspond to API major/minor

changes, as shown in the table above. Note that API versions and XML versions are interrelated, but are not necessarily equal to each other.

Network Services Manager allows for the possibility of a single XML schema version being spread across multiple XSD files, each mapped to a specific set of REST representations defined in a `<representation>` block. A special representation named DEFAULT is used if another representation does not take precedence. `<representation>` contains a `<schemaLocation>` element that points at the XSD file.

In Network Services Manager 5.0.2, there is a single DEFAULT representation type with a single XML namespace and XSD file.

Compatibility

Because we allow extensions to be made to the schema as revisions instead of major/minor version changes, both the client and server must follow the same rules on how to behave when the schema version is not optimal.

The XML namespace identifies the schema major/minor version being used, but certain minor changes can occur as schema revisions without affecting the major/minor version, and therefore the namespace.

Both the server and client must obey the rules for handing unrecognized content as described at http://www.xml.com/pub/a/2004/10/27/extend.html?page=2:

- **Must Ignore** rule: Document consumers MUST ignore any XML attributes or elements in a valid XML document that they do not recognize.
- **Must Ignore All** rule: The Must Ignore rule applies to unrecognized elements and their descendants in data-oriented formats.

In these rules, the word *ignore* means that these elements must be disregarded in processing if they are not understood. In situations where the message is modified and returned (such as doing a GET followed by a PUT), these ignored elements are left in place, unmodified.

1.10.3 When do API Versions Change?

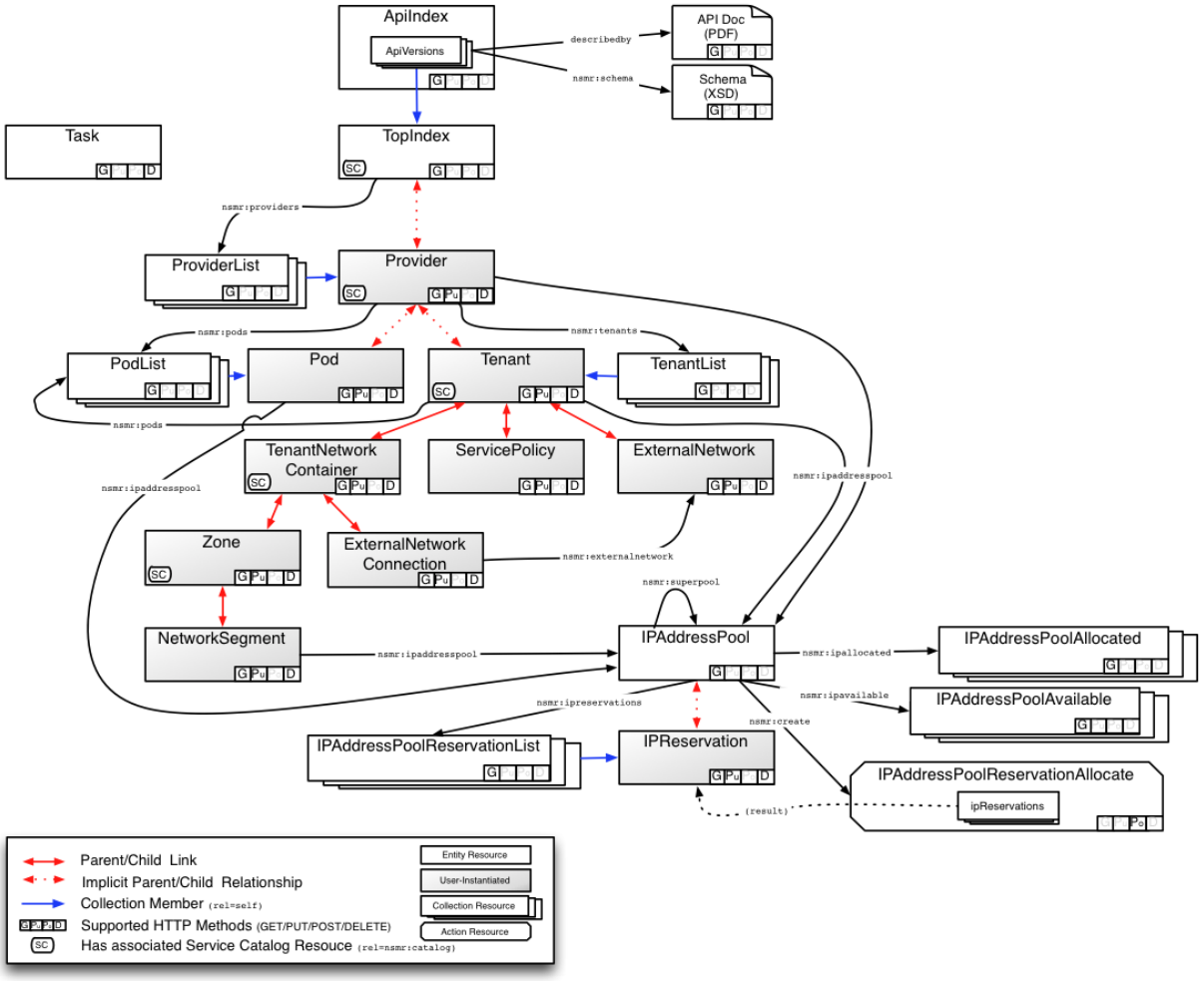
The API version and Network Services Manager version are not directly related, but as a general guideline, the following rules will be followed:

- Any version of Network Services Manager can increment the API major version, however:
    - It is likely that changes to the API major version will be held for major releases of Network Services Manager. If not, the prior major version will be supported until the next major Network Services Manager release.
    - If a new API version is introduced in what is considered a minor release of Network Services Manager, backward compatibility with the prior API major version is guaranteed (though certain features might require using the new API version) for at least one minor release.
- Any version of Network Services Manager can increment the API minor version. If this occurs, the client needs to be tested, and possibly modified, to support the new API version.

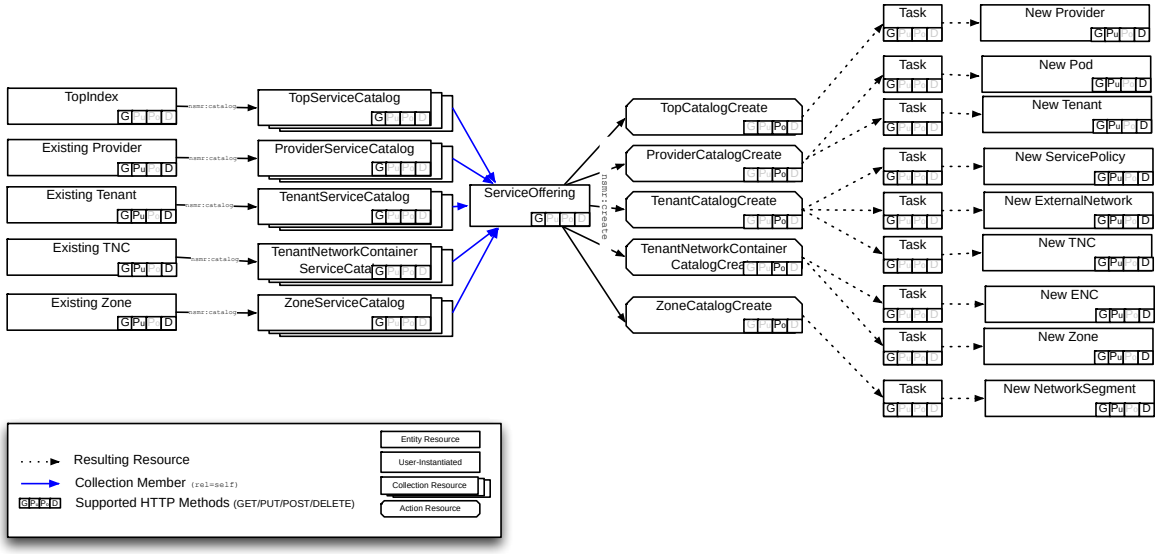
2 API Overview

2.1 Resource Diagrams

This overview diagram illustrates the key REST resources exposed in the Network Services Manager API and the relationships (links) between them:



Each resource that has an associated service catalog (indicated with the SC symbol) has its own collection of service offerings, which are used to create new resources as follows:



2.2 Call Summary

Operation	Request	Request Body	Synchronous Response	Asynchronous Response
API Index	`GET {API-URL}`	None	`200 OK` `<apiIndex>`	N/A
Top Index	`GET {API-URL}/v1`	None	`200 OK` `<topIndex>`	N/A
Get catalog of provider types	`GET {API-URL}/v1/top/catalog`	None	`200 OK` `<serviceCatalog>`	N/A

Create a Provider	`POST {API-URL}/v1/top/catalog/create`	`<provider>`	`202 Accepted`
			`<task>`
				`201 Created`
				`<provider>`
				`400 Bad request`
				`409 Conflict`

Get Provider Details	`GET {API-URL}/v1/provider/{provider-id}`	None	`200 OK` `<provider>`	N/A
List Providers	`GET {API-URL}/v1/top/provider`	None	`200 OK` `<providers>`	N/A
Get Service Catalog for Provider	`GET {API-URL}/v1/provider/{provider-id}/catalog`	None	`200 OK` `<serviceCatalog>`	N/A
List PODs for Provider	`GET {API-URL}/v1/provider/{provider-id}/pod`	None	`200 OK` `<pods>`	N/A

Create a POD	`POST {API-URL}/v1/provider/catalog/create`	`<serviceOffering>`	`202 Accepted`
			`<task>`
				`201 Created`
				`<pod>`
				`400 Bad request`
				`409 Conflict`

Update a POD	`PUT {API-URL}/v1/pod/{pod-id}`
	or
	`POST {API-URL}/v1/pod/{pod-id}/put`
		`<pod>`	`202 Accepted`
			`<task>`
				`200 OK`
				`<provider>`
				`400 Bad Request`
				`409 Conflict`
				`404 Not Found`

Delete a POD	DELETE {API-URL}/v1/pod/*{pod-id}* or POST {API-URL}/v1/pod/*{pod-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Update a Provider	PUT {API-URL}/v1/provider/*{provider-id}* or POST {API-URL}/v1/provider/*{provider-id}*/put	<provider>	202 Accepted <task>	200 OK <provider> 400 Bad Request 409 Conflict 404 Not Found
Delete a Provider	DELETE {API-URL}/v1/provider/*{provider-id}* or POST {API-URL}/v1/provider/*{provider-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Create a Tenant	POST {API-URL}/v1/provider/catalog/create	<serviceOffering>	202 Accepted <task>	201 Created <tenant> 400 Bad request 409 Conflict
Get Tenant Details	GET {API-URL}/v1/tenant/*{tenant-id}*	None	200 OK <tenant>	N/A
List tenants	GET {API-URL}/v1/{provider-id}/tenant/	None	200 OK <tenants>	N/A
Get Service Catalog for Tenant	GET {API-URL}/v1/tenant/*{tenant-id}*/catalog	None	200 OK <serviceCatalog>	N/A
Update a Tenant	PUT {API-URL}/v1/tenant/*{tenant-id}* or POST {API-URL}/v1/tenant/*{tenant-id}*/put	<tenant>	202 Accepted <task>	200 OK <tenant> 400 Bad Request 409 Conflict 404 Not Found
Delete a Tenant	DELETE {API-URL}/v1/tenant/*{tenant-id}* or POST {API-URL}/v1/tenant/*{tenant-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Create a Tenant Network Container (TNC)	POST {API-URL}/v1/tenant/*{tenant-id}*/catalog/create	<serviceOffering>	202 Accepted <task>	201 Created <tenantNetworkContainer> 400 Bad Request 409 Conflict
Get TNC Details	GET {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}*	None	200 OK <tenantNetworkContainer>	N/A
Get Service Catalog for TNC	GET {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}*/catalog	None	200 OK <serviceCatalog>	N/A

Update a TNC	PUT {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}* or POST {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}*/put	<tenantNetworkContainer>	202 Accepted <task>	200 OK <tenantNetworkContainer> 400 Bad Request 409 Conflict 404 Not Found
Delete a TNC	DELETE {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}* or POST {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Create an External Network	POST {API-URL}/v1/tenant/*{tenant-id}*/catalog/create	<serviceOffering>	202 Accepted <task>	201 Created <externalNetworkConnection> 400 Bad Request 409 Conflict
Get External Network Details	GET {API-URL}/v1/externalnetwork/*{en-id}*	None	200 OK <externalNetwork>	N/A
Update an External Network	PUT {API-URL}/v1/externalnetwork/*{en-id}* or POST {API-URL}/v1/externalnetwork/*{en-id}*/put	<externalNetwork>	202 Accepted <task>	200 OK <externalNetwork> 400 Bad Request 409 Conflict 404 Not Found
Delete an External Network	DELETE {API-URL}/v1/externalnetwork/*{en-id}* or POST {API-URL}/v1/externalnetwork/*{en-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Create an External Network Connection (ENC)	POST {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}*/catalog/create	<serviceOffering>	202 Accepted <task>	201 Created <externalNetworkConnection> 400 Bad Request 409 Conflict
Get ENC Details	GET {API-URL}/v1/externalnetworkconnection/*{enc-id}*	None	200 OK <externalNetworkConnection>	N/A
Update an ENC	PUT {API-URL}/v1/externalnetworkconnection/*{enc-id}* or POST {API-URL}/v1/externalnetworkconnection/*{enc-id}*/put	<externalNetworkConnection>	202 Accepted <task>	200 OK <externalNetworkConnection> 400 Bad Request 409 Conflict 404 Not Found
Delete an ENC	DELETE {API-URL}/v1/externalnetworkconnection/*{enc-id}* or POST {API-URL}/v1/externalnetworkconnection/*{enc-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found

Create a Zone	POST {API-URL}/v1/tenantnetworkcontainer/*{tnc-id}*/catalog/create	None	202 Accepted <task>	201 Created <zone> 400 Bad Request 409 Conflict
Get Zone Details	GET {API-URL}/v1/zone/*{zone-id}*	None	200 OK <zone>	N/A
Update a Zone	PUT {API-URL}/v1/zone/*{zone-id}* or POST {API-URL}/v1/zone/*{zone-id}*/put	<zone>	202 Accepted <task>	200 OK <zone> 400 Bad Request 409 Conflict 404 Not Found
Delete a Zone	DELETE {API-URL}/v1/zone/*{zone-id}* or POST {API-URL}/v1/zone/*{zone-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Create a NetworkSegment	POST {API-URL}/v1/zone/*{zone-id}*/catalog/create	None	202 Accepted <task>	201 Created <networkSegment> 400 Bad Request 409 Conflict
Get NetworkSegment Details	GET {API-URL}/v1/networksegment/*{networksegment-id}*	None	200 OK <networkSegment>	N/A
Update a NetworkSegment	PUT {API-URL}/v1/networksegment/*{networksegment-id}* or POST {API-URL}/v1/networksegment/*{networksegment-id}*/put	<networkSegment>	202 Accepted <task>	200 OK <networkSegment> 400 Bad Request 409 Conflict 404 Not Found
Delete a NetworkSegment	DELETE {API-URL}/v1/networksegment/*{networksegment-id}* or POST {API-URL}/v1/networksegment/*{networksegment-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Get IPAddressPool Details	GET {API-URL}/v1/ipaddresspool/*{ipaddresspool-id}*	None	200 OK <ipAddressPool>	N/A
Create IPReservation	POST {API-URL}/v1/ipaddresspool/*{ipaddresspool-id}*/reservation/allocate	<ipReservationRequest>	202 Accepted <task>	200 OK <ipReservationRequest> 400 Bad Request 409 Conflict 404 Not Found
Get IPReservation Details	GET {API-URL}/v1/ipreservation/*{ipreservation-id}*	None	200 OK <ipReservation>	N/A
Update IPReservation	PUT {API-URL}/v1/ipreservation/*{ipreservation-id}* or POST {API-URL}/v1/ipreservation/*{ipreservation-id}*/put	<ipReservation>	202 Accepted <task>	200 OK <ipReservation> 400 Bad Request 409 Conflict 404 Not Found

Delete IPReservation	DELETE {API-URL}/v1/ipreservation/*{ipreservation-id}* or POST {API-URL}/v1/ipreservation/*{ipreservation-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Create Firewall ServicePolicy	POST {API-URL}/v1/tenant/*tenant-001*/catalog/create	None	202 Accepted <task>	201 Created <servicePolicy> 400 Bad Request 409 Conflict
Create NAT ServicePolicy	POST {API-URL}/v1/tenant/*tenant-001*/catalog/create	None	202 Accepted <task>	201 Created <servicePolicy> 400 Bad Request 409 Conflict
Create LoadBalancer ServicePolicy	POST {API-URL}/v1/tenant/*tenant-001*/catalog/create	None	202 Accepted <task>	201 Created <servicePolicy> 400 Bad Request 409 Conflict
Get ServicePolicy Details	GET {API-URL}/v1/servicepolicy/{servicepolicy-id}	None	200 OK <servicePolicy>	N/A
Update ServicePolicy	PUT {API-URL}/v1/servicepolicy/*{servicepolicy-id}* or POST {API-URL}/v1/servicepolicy/*{servicepolicy-id}*/put	<servicePolicy>	202 Accepted <task>	200 OK <servicePolicy> 400 Bad Request 409 Conflict 404 Not Found
Delete ServicePolicy	DELETE {API-URL}/v1/servicepolicy/*{servicepolicy-id}* or POST {API-URL}/v1/servicepolicy/*{servicepolicy-id}*/delete	None	202 Accepted <task>	200 OK 404 Not Found
Retrieve Task	GET {API-URL}/v1/task/*{task-id}*	None	200 OK <task>	N/A
Delete Task	DELETE {API-URL}/v1/task/*{task-id}* or POST {API-URL}/v1/task/*{task-id}*/delete	None	200 OK 404 Not Found	N/A
Get Service Catalog Entry	GET {API-URL}/v1/serviceoffering/*{offering-id}*	None	200 OK <serviceOffering>	N/A

3 API Call Details

3.1 XML Samples

In all of the following XML examples, certain items have been omitted for clarity. In particular:

- All requests are assumed to include the proper XML namespace references:

```
xmlns:ns2="http://www.w3.org/2005/Atom"
xmlns:nsmr="http://www.cisco.com/NetworkServicesManager/rel"
xmlns="http://www.cisco.com/NetworkServicesManager"
```

- All HTTP operations must include appropriate HTTP basic-auth headers, as well as the `Accept: application/xml` and `Content-Type: application/xml` headers.
- In the examples, the identifiers used in URLs and UIDs are of the form "tenant-001" for readability. In reality, the identifiers are long hexadecimal UUID-style strings, resulting in URLs such as the following:

```
https://nsmhost.mydomain.com:8443/NetworkServicesManagerAPI/v1/service
offering/7F53F648FFA8010A01F217ECB671F0E8
```

- `<link>` elements in the examples often omit the optional "title" and attributes, but Network Services Manager will generally provide these. See the section Linked Resources for more details on these attributes.

3.2 Create a Provider

The highest level of resources that are exposed through the NBI hierarchy is *Providers*. The provider is defined outside of Network Services Manager and can represent an organization or individual that consumes cloud resources.

For purposes of this API, a Provider is an entity with a unique name. Registering the provider with the Network Services Manager will return a UID (unique ID) by which this provider will be known to this instance of Network Services Manager.

Most deployments require only a single type of provider, but the system has been designed to allow for the possibility of multiple types of providers. As a result, registering a provider is a two-step process:

1. A list of provider types is retrieved, and one chosen.
2. A request to create a provider of that type is submitted to the server.

Step 1: Retrieve Provider Type Catalog

3.2.1 Request

```
GET {API-URL}/v1/top/catalog
```

This top-level catalog object is linked from the API entry point, {API-URL}/v1/, and can be used to expose other types of creatable objects in the future. In the current release, it contains only ServiceOfferings of type "provider". Clients should check the offering type and ignore non-provider offerings, for future compatibility.

3.2.2 Request Body

None

3.2.3 Response Status Codes

```
200 OK – Catalog entries are returned
404 Not Found – Catalog is not found.
```

3.2.4 Response Body

```
<serviceOfferings>
  <serviceOffering type="provider">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-001</uid>
    <name>Service Provider</name>
    <description>The model to instantiate a service provider, that will prompt
for creation of a Global IP Address Pool, Infrastructure IP Address
Pool.</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-001" rel="self"
/>
      <link href="{API-URL}/v1/top/catalog/create" rel="nsmr:create" />
    </links>
    <parameters>
      <parameter>
        <name>global.pool</name>
        <type>subnets</type>
        <subnets/>
      </parameter>
    </parameters>
  </serviceOffering>
</serviceOfferings>
```

Step 2: Create Provider

3.2.5 Request

```
POST {API-URL}/v1/top/catalog/create
```

(This is the "create" link from the service offering)

3.2.6 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the Provider. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="provider">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-001</uid>
  <name>Default Provider</name>
  <description>Default Provider</description>
  <version>1</version>
  <parameters>
    <parameter>
      <name>global.pool</name>
      <type>subnets</type>
      <subnets>
        <subnet>
           <ipv4>10.1.0.0</ipv4>
           <mask>24</mask>
        </subnet>
      </subnets>
    </parameter>
  </parameters>
</serviceOffering>
```

3.2.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.2.8 Asynchronous Response Status Codes

201 Created – Provider has been created. Location header gives URL of created provider.
400 Bad Request – Something was wrong with the request, such as missing data
409 Conflict – Name conflicts with existing provider

3.2.9 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Provider Default Provider</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
   <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.2.10 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Provider Default Provider</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ProviderValue">
    <uid>{API-URL}/v1/provider/provider-001</uid>
    <name>Default Provider</name>
    <description>Default Provider</description>
    <version>1</version>
    <links>
      <link rel="self" href="{API-URL}/v1/provider/provider-001"/>
      <link rel="nsmr:pods" href="{API-URL}/v1/provider/provider-001/pod"/>
      <link rel="nsmr:catalog" href="{API-URL}/v1/provider/provider-
001/catalog"/>
    </links>
    <properties>
      <property>
        <name>global.pool</name>
        <subnets>
          <subnet>
             <ipv4>10.1.0.0</ipv4>
              <mask>24</mask>
          </subnet>
```

```
        </subnets>
      </property>
    </properties>
  </result>
</task>
```

3.3 Get Provider Details

Retrieve details about a previously-created provider entity.

3.3.1 Request

```
GET {API-URL}/v1/provider/{provider-id}
```

3.3.2 Request Body

None

3.3.3 Response Status Codes

```
200 OK – Provider detail
404 Not Found – Provider is not found.
```

3.3.4 Response Body

```
<provider>
  <uid>{API-URL}/v1/provider/provider-001</uid>
  <name>Default Provider</name>
  <description>Default Provider</description>
  <version>1</version>
  <links>
    <link rel="self" href="{API-URL}/v1/provider/provider-001"/>
    <link rel="parent" href="{API-URL}/v1"/>
    <link rel="nsmr:pods" href="{API-URL}/v1/provider/provider-001/pod"/>
    <link rel="nsmr:catalog" href="{API-URL}/v1/provider/provider-001/catalog"/>
    <link rel="nsmr:tenants" href="{API-URL}/v1/provider/provider-001/tenant"/>
  </links>
  <properties>
    <property>
      <name>global.pool</name>
      <subnets>
        <subnet>
           <ipv4>10.1.0.0</ipv4>
           <mask>24</mask>
        </subnet>
      </subnets>
    </property>
  </properties>
</provider>
```

3.4 List Providers

Returns a collection of providers that are known. The `<provider>` elements will contain a complete XML representation of a provider as shown in section 3.3.4.

3.4.1 Request

```
GET {API-URL}/v1/top/provider
```

3.4.2 Request Body

None

3.4.3 Response Status Codes

```
200 OK – Collection of provider objects are returned.
```

3.4.4 Response Body

```
<providers>
  <provider>
    <uid>{API-URL}/v1/provider/provider-001</uid>
    <name>Default Provider</name>
    <links>
      <link rel="self" href="{API-URL}/v1/provider/provider-001"/>
    </links>
  </provider>
</providers>
```

3.5 Get Service Catalog for Provider

A "Service Catalog" is a collection of "Service Catalog Entries" (or "Service Offerings") that identify services that can be created within some context. Any service offering in the catalog can then be requested.

In Network Services Manager 5.0.2, Provider Service catalogs can contain the following service offering types:

- pod
- tenant

3.5.1 Request

```
GET {API-URL}/v1/provider/provider-001/catalog
```

This URL is the "`nsmr:catalog`" link of the Provider resource.

3.5.2 Request Body

None

3.5.3 Response Status Codes

```
200 OK - Catalog entries are returned
404 Not Found - Provider is not found.
```

3.5.4 Response Body

```
<serviceOfferings>
  <serviceOffering type="tenant">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-002</uid>
    <name>Four Zone Tenant</name>
    <description>A model for a tenant environment with four zones,
      Internet Edge, Secured Internet Edge, Private Edge and Secured Private
Edge.
      Internet Edge Zone has an Internet RAP associated with it, and the Private
Edge
      has a Private RAP associated with it. All four zones are connected to
      each other using a shared Firewall context. Each tenant shares the
      Global IP Address Pool and has his own Routable Private IP Address Pool.
These
      are defined on tenant creation. The maximum number of tenants supported
      by this model can be set with the limit attribute.
    </description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-002" rel="self"
/>
      <link href="{API-URL}/v1/provider/provider-001/catalog/create"
rel="nsmr:create" />
    </links>
    <parameters>
```

```
      <parameter>
        <name>routable.private.address.pool</name>
        <type>subnets</type>
        <subnets>
          <subnet>
            <ipv4>192.168.240.0</ipv4>
            <mask><20/mask>
          </subnet>
        </subnets>
      </parameter>
    </parameters>
  </serviceOffering>
  <serviceOffering type="pod">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-003</uid>
    <name>VMDC 2.1 Compact POD with Edge Routers</name>
    <description></description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-003" rel="self"
/>
      <link href="{API-URL}/v1/provider/provider-003/catalog/create"
rel="nsmr:create" />
    </links>
    <parameters>
      [omitted for space]
    </parameters>
  </serviceOffering>
</serviceOfferings>
```

3.6 List PODs for Provider

A collection of the PODs (deployment sites for TNCs) that are available to tenants within a particular provider. The `<pod>` elements will contain a complete XML representation of a provider as shown in section 3.7.

3.6.1 Request

```
GET {API-URL}/v1/provider/provider-001/pod
```

This URL is the "`nsmr:pods`" link of the Provider resource.

3.6.2 Request Body

None

3.6.3 Response Status Codes

```
200 OK - POD entries are returned.
404 Not Found - Provider is not found.
```

3.6.4 Response Body

```
<pods>
  <pod>
    <uid>{API-URL}/v1/pod/pod-001</uid>
    <name>San Jose POD 01</name>
    <description>San Jose POD 01, SJC12-1</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/pod/pod-001" rel="self" />
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    </links>
    <properties>
       <!-- all properties, omitted for space -->
    </properties>
```

```
  </pod>
  <pod>
    <uid>{API-URL}/v1/pod/pod-002</uid>
    <name>Denver</name>
    <description>Another one.</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/pod/pod-002" rel="self" />
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    </links>
    <properties>
       <!-- all properties, omitted for space -->
    </properties>
</pods>
```

3.7 Create/Update/Delete a POD

Network Services Manager provides an API for registering, modifying, and removing PODs. `POD` Service Offerings are found in the provider's service catalog. (See Get Service Catalog For Provider for details.)

A given Network Services Manager instance can support multiple predefined flavors of POD. They will be listed in the Provider's service catalog as ServiceOfferings with a type of "`pod`".

The specific parameters will vary depending on the POD flavor, but will include names, addresses, and credentials for each required device in the POD.

These parameters can be modified via the normal update mechanism. You can modify only the parameters that are supported by the POD object. You cannot customize the POD design by changing its network elements (devices) or interconnects because these are statically defined.

3.8 Update a Provider

3.8.1 Request

`PUT {API-URL}/v1/provider/`*`provider-001`* or
`POST {API-URL}/v1/provider/`*`provider-001`*`/put`

3.8.2 Request Body

```
<provider>
  <uid>{API-URL}/v1/provider/provider-001</uid>
  <name>Default Provider</name>
  <description>Default Provider New Description</description>
  <version>1</version>
  <links>
    <link rel="self" href="{API-URL}/v1/provider/provider-001"/>
    <link rel="parent" href="{API-URL}/v1"/>
    <link rel="nsmr:pods" href="{API-URL}/v1/provider/provider-001/pod"/>
    <link rel="nsmr:catalog" href="{API-URL}/v1/provider/provider-001/catalog"/>
    <link rel="nsmr:tenants" href="{API-URL}/v1/provider/provider-001/tenant"/>
  </links>
  <properties>
    <property>
      <name>global.pool</name>
      <subnets>
        <subnet>
           <ipv4>10.1.0.0</ipv4>
           <mask>24</mask>
        </subnet>
      </subnets>
    </property>
```

```
  </properties>
</provider>
```

3.8.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.8.4 Asynchronous Response Status Codes

200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Provider was not found.
409 Conflict – Updating the Provider would cause some conflict (for instance, name already in use, version number mismatch).

3.8.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Provider provider-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.8.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Provider provider-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ProviderValue">
    <uid>{API-URL}/v1/provider/provider-001</uid>
    <name>Default Provider</name>
    <description>Default Provider New Description</description>
    <version>1</version>
    <links>
      <link rel="self" href="{API-URL}/v1/provider/provider-001"/>
      <link rel="parent" href="{API-URL}/v1"/>
      <link rel="nsmr:pods" href="{API-URL}/v1/provider/provider-001/pod"/>
      <link rel="nsmr:catalog" href="{API-URL}/v1/provider/provider-
001/catalog"/>
      <link rel="nsmr:tenants" href="{API-URL}/v1/provider/provider-
001/tenant"/>
    </links>
    <properties>
```

```
      <property>
        <name>global.pool</name>
        <subnets>
          <subnet>
             <ipv4>10.1.0.0</ipv4>
             <mask>24</mask>
          </subnet>
        </subnets>
      </property>
    </properties>
  </result>
</task>
```

3.8.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.8.8 Asynchronous Response Status Codes

200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Provider was not found.
409 Conflict – For example, version number does not match.

3.9 Delete a Provider

3.9.1 Description

Remove a Provider from this instance of Network Services Manager.

3.9.2 Request

DELETE {API-URL}/v1/provider/*provider-001* or
POST {API-URL}/v1/provider/*provider-001*/delete

This URL is the "self" link of the representation of the provider.

3.9.3 Request Body

None

3.9.4 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.9.5 Asynchronous Response Status Codes

200 OK – Provider has been deleted.
404 Not Found – Provider was not found.

3.9.6 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Provider provider-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
```

```
  <endTime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.9.7 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Provider provider-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
   <taskStatus>success</taskStatus>
   <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete
</task>
```

3.10 Create a Tenant

A tenant is a logical owner of one or more network containers. The tenant is defined outside of Network Services Manager and can represent an organization or individual that consumes cloud resources.

For purposes of this API, a Tenant is an entity with a unique name. Registering the tenant with will return a UID (unique ID) by which this tenant will be known to this instance of Network Services Manager.

It is expected that most deployments will only require a single type of tenant, but the system has been designed to allow for the possibility of multiple types of tenants.

Because of this, registering a tenant is a two-step process. First, a list of tenant types is retrieved, and one chosen. Then a request to create a tenant of that type is submitted to the server.

Tenant Service Offerings are found in the provider's service catalog. (See Get Service Catalog for Provider.)

Request

```
POST {API-URL}/v1/provider/provider-001/catalog/create
```

(This is the "create" link from the service offering)

Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the Tenant. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="tenant">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-002</uid>
  <name>Tenant01</name>
  <description>Tenant01</description>
  <version>1</version>
  <parameters>
    <parameter>
      <name>routable.private.address.pool</name>
      <type>subnets</type>
```

```
      <subnets>
        <subnet>
          <ipv4>192.168.240.0</ipv4>
          <mask>20</mask>
        </subnet>
      </subnets>
    </parameter>
  </parameters>
</serviceOffering>
```

Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

Asynchronous Response Status Codes

201 Created – Tenant has been created. Location header gives URL of created tenant.
400 Bad Request – Something was wrong with the request, such as missing data
409 Conflict – Name conflicts with existing tenant

Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Tenant Tenant01</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
   <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Tenant Tenant01</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="TenantValue">
    <uid>{API-URL}/v1/tenant/tenant-001</uid>
    <name>Tenant01</name>
    <description>Tenant01</description>
    <version>1</version>
    <links>
      <link rel="self" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
```

```
      <link creatrel="nsmr:catalog" href="{API-URL}/v1/tenant/tenant-
001/catalog"/>
    </links>
    <properties>
      <property>
        <name>routable.private.address.pool</name>
        <subnets>
          <subnet>
            <ipv4>192.168.240.0</ipv4>
            <mask>20</mask>
          </subnet>
        </subnets>
      </property>
    </property>
  </result>
</task>
```

3.11 Get Tenant Details

Retrieve details about a previously-created tenant entity.

3.11.1 Request

```
GET {API-URL}/v1/tenant/{tenant-id}
```

3.11.2 Request Body

None

3.11.3 Response Status Codes

```
200 OK - Tenant detail
404 Not Found - Tenant is not found.
```

3.11.4 Response Body

```
<tenant>
  <uid>{API-URL}/v1/tenant/tenant-001</uid>
  <name>Tenant01</name>
  <description>Tenant01</description>
  <version>1</version>
  <links>
    <link rel="self" href="{API-URL}/v1/tenant/tenant-001"/>
    <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    <link rel="nsmr:catalog" href="{API-URL}/v1/tenant/tenant-001/catalog"/>
    <link rel="child" href="{API-URL}/v1/tenantnetworkcontainer/tnc-001"/>
  </links>
  <properties>
    <property>
      <name>routable.private.address.pool</name>
      <subnets>
        <subnet>
          <ipv4>192.168.240.0</ipv4>
          <mask>20</mask>
        </subnet>
      </subnets>
    </property>
  </properties>
</tenant>
```

NOTE

After a TNC, ServicePolicy, or ExternalNetwork is created under a Tenant, getting the Tenant details will include child links to the created entities. This mechanism enables "walking" through child links and retrieving whatever elements were created via this API.

3.12 List Tenants

Returns a collection of tenants that are known. The `<tenant>` elements will contain a complete XML representation of a provider as shown in section 3.11.4.

3.12.1 Request

```
GET {API-URL}/v1/provider/provider-001/tenant/
```

3.12.2 Request Body

None

3.12.3 Response Status Codes

```
200 OK – Collection of tenant objects are returned.
```

3.12.4 Response Body

```
<tenants>
  <tenant>
    <uid>{API-URL}/v1/tenant/tenant-001</uid>
    <name>Tenant01</name>
    <links>
      <link rel="self" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    </links>
  </tenant>
  <tenant>
    <uid>{API-URL}/v1/tenant/tenant-002</uid>
    <name>Tenant02</name>
    <links>
      <link rel="self" href="{API-URL}/v1/tenant/tenant-002"/>
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    </links>
  </tenant>
</tenants>
```

3.13 Get Service Catalog for Tenant

A "Service Catalog" is a collection of "Service Catalog Entries" (or "Service Offerings") that identify services that can be created within some context. Any service offering in the catalog can then be requested.

In Network Services Manager 5.0.2, Tenant Service catalogs can contain the following service offering types:

- servicepolicy
- externalnetwork
- tenantnetworkcontainer

3.13.1 Request

```
GET {API-URL}/v1/tenant/tenant-001/catalog
```

This URL is the "`nsmr:catalog`" link of the Tenant resource.

3.13.2 Request Body

None

3.13.3 Response Status Codes

```
200 OK – Catalog entries are returned
404 Not Found – Tenant is not found.
```

3.13.4 Response Body

```
<serviceOfferings>
  <serviceOffering type="tenantnetworkcontainer">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-004</uid>
    <name>Tenant Network Container</name>
    <description>The Tenant Network Container is assigned to an NSM
POD</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-004" rel="self"
/>
      <link href="{API-URL}/v1/tenant/tenant-001/catalog/create"
rel="nsmr:create" />
    </links>
    <parameters>
      <parameter>
        <name>tnc.pod</name>
        <type>uid</type>
        <uid/>
      </parameter>
    </parameters>
  </serviceOffering>
  <serviceOffering type="servicepolicy">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-005</uid>
    <name>Firewall Service</name>
    <description>This Service will allow for creation of firewall rules in a
context.</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-005" rel="self"
/>
      <link href="{API-URL}/v1/tenant/tenant-001/catalog/create"
rel="nsmr:create" />
    </links>
    <parameters>
      <parameter>
        <name>traffic.src</name>
        <type>trafficFilters</type>
        <trafficfilters/>
      </parameter>
      <parameter>
        <name>traffic.dst</name>
        <type>trafficFilters</type>
        <trafficfilters/>
      </parameter>
      <parameter>
        <name>fw.service.ports</name>
        <type>protocolServices</type>
        <protocolServices/>
      </parameter>
    </parameters>
  </serviceOffering>
  <serviceOffering type="externalnetwork">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-006</uid>
    <name>External Network</name>
```

```
    <description>A set of external networks to be used by the ENC or by Service
Policies</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-006" rel="self"
/>
      <link href="{API-URL}/v1/tenant/tenant-001/catalog/create"
rel="nsmr:create" />
    </links>
    <parameters>
      <parameter>
        <name>external.subnets</name>
        <type>subnets</type>
        <subnets>
          <subnet>
            <ipv4>0.0.0.0</ipv4>
            <mask>0</mask>
          </subnet>
        </subnets>
      </parameter>
    </parameters>
  </serviceOffering>
</serviceOfferings>
```

3.14 List PODs available to a Tenant

A collection of PODs (deployment sites for TNCs) that are available to a specific tenant (those provided by the provider within which the Tenant was created). The `<pod>` elements will contain a complete XML representation of a provider as shown in section 3.7.

3.14.1 Request

`GET {API-URL}/v1/tenant/`*`tenant-001`*`/pod`

This URL is the "`nsmr:pods`" link of the Tenant resource.

3.14.2 Request Body

None

3.14.3 Response Status Codes

```
200 OK - POD entries are returned.
404 Not Found - Provider is not found.
```

3.14.4 Response Body

```
<pods>
  <pod>
    <uid>{API-URL}/v1/pod/pod-001</uid>
    <name>San Jose POD 01</name>
    <description>San Jose POD 01, SJC12-1</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/pod/pod-001" rel="self" />
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    </links>
    <properties>
       <!-- all properties, omitted for space -->
    </properties>
  </pod>
  <pod>
    <uid>{API-URL}/v1/pod/pod-002</uid>
    <name>Denver</name>
    <description>Another one.</description>
```

```
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/pod/pod-002" rel="self" />
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    </links>
    <properties>
       <!-- all properties, omitted for space -->
    </properties>
</pods>
```

3.15 Update a Tenant

3.15.1 Request

PUT {API-URL}/v1/tenant/*tenant-001* or
POST {API-URL}/v1/tenant/*tenant-001*/put

3.15.2 Request Body

```
<tenant>
  <uid>{API-URL}/v1/tenant/tenant-001</uid>
  <name>Tenant01</name>
  <description>Tenant01 New Description</description>
  <version>1</version>
  <links>
    <link rel="self" href="{API-URL}/v1/tenant/tenant-001"/>
    <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
    <link rel="nsmr:catalog" href="{API-URL}/v1/tenant/tenant-001/catalog"/>
    <link rel="child" href="{API-URL}/v1/tenantnetworkcontainer/tnc-001"/>
  </links>
  <properties>
    <property>
      <name>routable.private.address.pool</name>
      <subnets>
        <subnet>
          <ipv4>192.168.240.0</ipv4>
          <mask>20</mask>
        </subnet>
      </subnets>
    </property>
  </properties>
</tenant>
```

3.15.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.15.4 Asynchronous Response Status Codes

200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Tenant was not found.
409 Conflict – Updating the Tenant would cause some conflict (for instance, name already in use, version number mismatch).

3.15.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Tenant tenant-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
```

```
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.15.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Tenant tenant-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="TenantValue">
    <uid>{API-URL}/v1/tenant/tenant-001</uid>
    <name>Tenant01</name>
    <description>Tenant01 New Description</description>
    <version>2</version>
    <links>
      <link rel="self" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="parent" href="{API-URL}/v1/provider/provider-001"/>
      <link rel="nsmr:catalog" href="{API-URL}/v1/tenant/tenant-001/catalog"/>
      <link rel="child" href="{API-URL}/v1/tenantnetworkcontainer/tnc-001"/>
    </links>
    <properties>
      <property>
        <name>routable.private.address.pool</name>
        <subnets>
          <subnet>
            <ipv4>192.168.240.0</ipv4>
            <mask>20</mask>
          </subnet>
        </subnets>
      </property>
    </properties>
  </result>
</task>
```

3.15.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.15.8 Asynchronous Response Status Codes

200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Tenant was not found.
409 Conflict – For example, version number does not match.

3.16 Delete a Tenant

3.16.1 Description

Remove a Tenant from this instance of Network Services Manager.

3.16.2 Request

```
DELETE {API-URL}/v1/tenant/tenant-001 or
POST {API-URL}/v1/tenant/tenant-001/delete
```

This URL is the "self" link of the representation of the tenant.

3.16.3 Request Body

None

3.16.4 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.16.5 Asynchronous Response Status Codes

```
200 OK – Tenant has been deleted.
404 Not Found – Tenant was not found.
```

3.16.6 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Tenant tenant-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.16.7 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Tenant tenant-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
   <taskStatus>success</taskStatus>
   <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.17 Create a Tenant Network Container (TNC)

Create a Tenant Network Container entity, based on a Service Offerings in the tenant entity catalog. The target POD is a required parameter for all TNC Service Offerings.

Tenant Network Container Service Offerings are found in the tenant's service catalog. (See Get Service Catalog for Tenant for details.)

The required parameter '`tnc.pod`' specifies the POD in which this TNC will be created. To determine the UID to pass for this parameter, the POD list can be consulted (See List PODs for Provider for details)

3.17.1 Request

```
POST {API-URL}/v1/tenant/tenant-001/catalog/create
```

3.17.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the TNC. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="tenantnetworkcontainer">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-004</uid>
  <name>Bill's TNC</name>
  <description>This TNC is for testing.</description>
  <version>1</version>
  <parameters>
    <parameter>
      <name>tnc.pod</name>
      <type>uid</type>
      <uid>{API-URL}/v1/pod/pod-001</uid>
    </parameter>
  </parameters>
</serviceOffering>
```

3.17.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.17.4 Asynchronous Response Status Codes

```
201 Created – TNC has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the TNC would cause some conflict (for instance, name
already in use).
```

3.17.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create TNC tnc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
```

```
  <percentComplete>0</percentComplete>
</task>
```

3.17.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create TNC tnc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="TenantNetworkContainerValue">
    <uid>{API-URL}/v1/tnc/tnc-001</uid>
    <name>Bill's TNC</name>
    <description>This TNC is for testing.</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="self" />
      <link href="{API-URL}/v1/tenant/tenant-001" rel="parent" />
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog"
rel="nsmr:catalog"/>
    </links>
    <properties>
      <property>
        <name>tnc.pod</name>
        <uid>{API-URL}/v1/pod/pod-001</uid>
      </property>
    </properties>
  </result>
</task>
```

3.18 Get TNC Details

Retrieve details about a previously-created Tenant Network Container entity.

3.18.1 Request

```
GET {API-URL}/v1/tenantnetworkcontainer/{tnc-id}
```

3.18.2 Request Body

None

3.18.3 Response Status Codes

```
200 OK - TNC detail
404 Not Found - TNC is not found.
```

3.18.4 Response Body

```
<tenantNetworkContainer>
  <uid>{API-URL}/v1/tnc/tnc-001</uid>
  <name>Bill's TNC</name>
  <description>This TNC is for testing.</description>
  <version>1</version>
  <links>
```

```
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="self" />
    <link href="{API-URL}/v1/tenant/tenant-001" rel="parent" />
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog"
rel="nsmr:catalog"/>
    <link rel="child" href="{API-URL}/v1/zone/zone-001"/>
  </links>
  <properties>
    <property>
      <name>tnc.pod</name>
      <uid>{API-URL}/v1/pod/pod-001</uid>
    </property>
  </properties>
</tenantNetworkContainer>
```

NOTE

After a Zone or ExternalNetworkConnection is created under a TNC, getting the TNC details will include child links to the created entities. This mechanism enables "walking" through child links and retrieving whatever elements were created via this API.

3.19 Get Service Catalog for TNC

A TNC has its own Service Catalog that lists the items that can be created within it. There is a link in the representation of the TNC to access it.

In Network Services Manager 5.0.2, Tenant Network Container Service catalogs can contain the following service offering types:

- zone
- externalnetworkconnection

3.19.1 Request

```
GET {API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog
```

This URL is the "`nsmr:catalog`" link of the Tenant Network Container resource.

3.19.2 Request Body

None

3.19.3 Response Status Codes

```
200 OK - Catalog entries are returned.
404 Not Found - TNC was not found.
```

3.19.4 Response Body

```
<serviceOfferings>
  <serviceOffering type="zone">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-005</uid>
    <name>Internet Edge</name>
    <description>This is an Internet Edge Zone and will be reachable directly
from the Internet</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-005" rel="self"
/>
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog/create"
rel="nsmr:create">
    </links>
    <parameters/>
  </serviceOffering>
  <serviceOffering type="externalnetworkconnection">
```

```
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-006</uid>
    <name>External Connection</name>
    <description>An external network connection describes the connection
strategy from remote locations to a zone</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-006" rel="self"
/>
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog/create"
rel="nsmr:create">
    </links>
    <parameters>
      <parameter>
        <type>uids</type>
        <name>external.networks</name>
        <uids/>
      </parameter>
    </parameters>
  </serviceOffering>
</serviceOfferings>
```

3.20 Update a TNC

3.20.1 Request

```
PUT {API-URL}/v1/tenantnetworkcontainer/tnc-001 or
POST {API-URL}/v1/tenantnetworkcontainer/tnc-001/put
```

3.20.2 Request Body

```
<tenantNetworkContainer>
  <uid>{API-URL}/v1/tnc/tnc-001</uid>
  <name>Bill's TNC</name>
  <description>This TNC is for testing!</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="self" />
    <link href="{API-URL}/v1/tenant/tenant-001" rel="parent" />
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog"
rel="nsmr:catalog"/>
    <link rel="child" href="{API-URL}/v1/zone/zone-001"/>
  </links>
  <properties>
    <property>
      <name>tnc.pod</name>
      <uid>{API-URL}/v1/pod/pod-001</uid>
    </property>
  </properties>
</tenantNetworkContainer>
```

3.20.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.20.4 Asynchronous Response Status Codes

```
200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – TNC was not found.
409 Conflict – Updating the TNC would cause some conflict (for instance, name
already in use, version number mismatch).
```

3.20.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update TNC tnc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.20.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update TNC tnc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="TenantNetworkContainerValue">
    <uid>{API-URL}/v1/tnc/tnc-001</uid>
    <name>Bill's TNC</name>
    <description>This TNC is for testing!</description>
    <version>2</version>
    <links>
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="self" />
      <link href="{API-URL}/v1/tenant/tenant-001" rel="parent" />
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog"
rel="nsmr:catalog"/>
      <link rel="child" href="{API-URL}/v1/zone/zone-001"/>
    </links>
    <properties>
      <property>
        <name>tnc.pod</name>
        <uid>{API-URL}/v1/pod/pod-001</uid>
      </property>
    </properties>
 </result>
</task>
```

3.20.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.20.8 Asynchronous Response Status Codes

200 OK – Update succeeded
400 Bad Request – There was an error in the input.

```
404 Not Found – TNC was not found.
409 Conflict – For example, version number does not match.
```

3.21 Delete a TNC

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.21.1 Request

```
DELETE {API-URL}/v1/tenantnetworkcontainer/tnc-001 or
POST {API-URL}/v1/tenantnetworkcontainer/tnc-001/delete
```

3.21.2 Request Body

None

3.21.3 Response Status Codes

```
200 OK – Delete succeeded
404 Not Found – TNC was not found.
```

3.21.4 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete TNC tnc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
   <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.21.5 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete TNC tnc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.22 Create an ExternalNetwork

Created at the level of a tenant, and referred to by External Network Connections (ENCs) which are later declared.

External Network Service Offerings are found in the Tenant's service catalog. (See Get Service Catalog For Tenant for details.)

3.22.1 Request

```
POST {API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog/create
```

3.22.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ExternalNetwork. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="externalnetwork">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-006</uid>
  <name>External Network</name>
  <description>My external network</description>
  <version>1</version>
  <parameters>
    <parameter>
      <name>external.subnets</name>
      <type>subnets</type>
      <subnets>
        <subnet>
          <ipv4>192.168.1.0</ipv4>
          <mask>24</mask>
        </subnet>
      </subnets>
    </parameter>
  </parameters>
</serviceOffering>
```

3.22.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.22.4 Asynchronous Response Status Codes

```
201 Created – ExternalNetwork has been created. Location header gives canonical
URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ExternalNetwork would cause some conflict (for
instance, name already in use).
```

3.22.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ExternalNetwork en-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
```

```
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.22.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ExternalNetwork en-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ExternalnetworkValue">
    <uid>{API-URL}/v1/externalnetwork/en-001</uid>
    <name>External Network</name>
    <description>My external network</description>
    <version>1</version>
    <properties>
      <property>
        <name>external.subnets</name>
        <type>subnets</type>
        <subnets>
          <subnet>
            <ipv4>192.168.1.0</ipv4>
            <mask>24</mask>
          </subnet>
        </subnets>
      </property>
    </properties>
  </result>
</task>
```

3.23 Get ExternalNetwork Details

Retrieve details about a previously-created Externalnetwork entity.

3.23.1 Request

```
GET {API-URL}/v1/externalnetwork/{externalnetwork-id}
```

3.23.2 Request Body

None

3.23.3 Response Status Codes

```
200 OK - Externalnetwork detail
404 Not Found - Externalnetwork is not found.
```

3.23.4 Response Body

```
<externalnetwork>
  <uid>{API-URL}/v1/externalnetwork/en-001</uid>
  <name>External Network</name>
  <description>My external network</description>
  <version>1</version>
```

```
  <links>
    <link href="{API-URL}/v1/externalnetwork/en-001" rel="self" />
  </links>
  <properties>
    <property>
      <name>external.subnets</name>
      <type>subnets</type>
      <subnets>
        <subnet>
          <ipv4>192.168.1.0</ipv4>
          <mask>24</mask>
        </subnet>
      </subnets>
    </property>
  </properties>
</externalnetwork>
```

3.24 Update an ExternalNetwork

3.24.1 Request

```
PUT {API-URL}/v1/externalnetwork/en-001 or
POST {API-URL}/v1/externalnetwork/en-001/put
```

3.24.2 Request Body

```
<externalnetwork>
  <uid>{API-URL}/v1/externalnetwork/en-001</uid>
  <name>External Network</name>
  <description>My external network 2</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/externalnetwork/en-001" rel="self" />
  </links>
  <properties>
    <property>
      <name>external.subnets</name>
      <type>subnets</type>
      <subnets>
        <subnet>
          <ipv4>192.168.1.0</ipv4>
          <mask>24</mask>
        </subnet>
      </subnets>
    </property>
  </properties>
</externalnetwork>
```

3.24.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.24.4 Asynchronous Response Status Codes

```
200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – ENC was not found.
409 Conflict – Updating the ENC would cause some conflict (for instance, name
already in use, version number mismatch).
```

3.24.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Externalnetwork en-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.24.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Externalnetwork en-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ExternalnetworkValue">
    <externalnetwork>
      <uid>{API-URL}/v1/externalnetwork/en-001</uid>
      <name>External Network</name>
      <description>My external network 2</description>
      <version>2</version>
      <links>
        <link href="{API-URL}/v1/externalnetwork/en-001" rel="self" />
      </links>
      <properties>
        <property>
          <name>external.subnets</name>
          <type>subnets</type>
          <subnets>
            <subnet>
              <ipv4>192.168.1.0</ipv4>
              <mask>24</mask>
            </subnet>
          </subnets>
        </property>
      </properties>
    </externalnetwork>
  </result>
</task>
```

3.24.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.24.8 Asynchronous Response Status Codes

```
200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Externalnetwork was not found.
409 Conflict – For example, version number does not match.
```

3.25 Delete an ExternalNetwork

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.25.1 Request

```
DELETE {API-URL}/v1/externalnetwork/en-001 or
POST {API-URL}/v1/externalnetwork/en-001/delete
```

3.25.2 Request Body

None

3.25.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.25.4 Asynchronous Response Status Codes

```
200 OK – Delete succeeded
404 Not Found – Externalnetwork was not found.
```

3.25.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Externalnetwork en-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.25.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Externalnetwork en-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
```

```
  <percentComplete>100</percentComplete>
</task>
```

3.26 Create an ExternalNetworkConnection (ENC)

External Network Connection Service Offerings are found in the Tenant Network Container service catalog. (See Get Service Catalog for TNC for details.)

To create an ENC you POST a request to the "create" link for the ENC ServiceOffering. The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous provisioning. The provided link can be used to retrieve the task details.

An ENC establishes a connection from a TNC to an external network. Therefore, when creating an ENC, the client must specify the ExternalNetwork (which you have already created) to which it will be connected.

The following parameters are required:

- `external.networks`: specifies one or more external networks to which this ENC will be attached. We recommend that clients store the UIDs of external networks as they are created so that they can be passed in here; however, if the UIDs are not stored, they can be discovered by consulting the "child" links of the Tenant object.
- `assigned.zone`: identifies the UID of the zone within the TNC to which this ENC is assigned.

After it is created, each ENC will provide the following two read-only properties:

Name	Type	Example

vrf	`vrf`	`<property>`
		`  <name>vrf</name>`
		`  <vrf>`
		`  <name>8a6fcd2</name>`
		`  <devices>`
		`  <vrfDevice>`
		`  <ipv4>10.86.241.19</ipv4>`
		`  </vrfDevice>`
		`  <vrfDevice>`
		`  <ipv4>10.86.241.20</ipv4>`
		`  </vrfDevice>`
		`  </devices>`
		`  </vrf>`
		`</property>`

routing	`routingStrategy`	`<property>`
		`  <name>routing</name>`
		`  <routingStrategy>`
		`  <ospf>`
		`  <area>1234</area>`
		`  <devices>`
		`  <routingDevice>`
		`  <ipv4>10.86.241.19</ipv4>`
		`  <process>4</process>`
		`  </routingDevice>`
		`  <routingDevice>`
		`  <ipv4>10.86.241.20</ipv4>`
		`  <process>4</process>`
		`  </routingDevice>`
		`  </devices>`
		`  </ospf>`
		`  </routingStrategy>`
		`</property>`

These attributes describe the layer-3 switches upon which the VRF for this ExternalNetwork was created and the specific routing details, for use by a northbound system that wishes to manage the provider edge device itself, rather than having Network Services Manager manage the provider edge.

3.26.1 Request

```
POST {API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog/create
```

3.26.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ENC. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="externalnetworkconnection">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-007</uid>
  <name>External Connection</name>
  <description>An external network</description>
  <version>1</version>
  <parameters>
    <parameter>
      <name>external.networks</name>
      <type>uids</type>
      <uids>
        <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
      </uids>
    </parameter>
    <parameter>
      <name>assigned.zone</name>
      <type>uid</type>
      <uid>{API-URL}/v1/zone/zone-001</uid>
    </parameter>
  </parameters>
</serviceOffering>
```

3.26.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.26.4 Asynchronous Response Status Codes

```
201 Created – ENC has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ENC would cause some conflict (for instance, name
already in use).
```

3.26.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ENC enc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
```

```
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.26.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ENC enc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ExternalNetworkConnectionValue">
    <uid>{API-URL}/v1/externalnetworkconnection/enc-001</uid>
    <name>External Connection</name>
    <description>An external network</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/externalnetworkconnection/enc-001" rel="self" />
      <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="parent" />
    </links>
    <properties>
      <property>
        <name>external.networks</name>
        <type>uids</type>
        <uids>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
        </uids>
      </property>
      <property>
        <name>assigned.zone</name>
        <type>uid</type>
        <uid>{API-URL}/v1/zone/zone-001</uid>
      </property>
      <property isReadOnly="true">
        <name>vrf</name>
        <vrf>
          <name>8a6fcd2</name>
          <devices>
            <vrfDevice>
              <ipv4>10.86.241.19</ipv4>
            </vrfDevice>
            <vrfDevice>
              <ipv4>10.86.241.20</ipv4>
            </vrfDevice>
          </devices>
        </vrf>
      </property>
      <property isReadOnly="true">
        <name>routing</name>
        <routingStrategy>
          <ospf>
            <area>1234</area>
            <devices>
              <routingDevice>
```

```
                <ipv4>10.86.241.19</ipv4>
                <process>4</process>
              </routingDevice>
              <routingDevice>
                <ipv4>10.86.241.20</ipv4>
                <process>4</process>
              </routingDevice>
            </devices>
          </ospf>
        </routingStrategy>
      </property>
    </properties>
  </result>
</task>
```

3.27 Get ExternalNetworkConnection Details

Retrieve details about a previously-created ExternalNetworkConnection entity.

3.27.1 Request

```
GET {API-URL}/v1/externalnetworkconnection/{externalnetworkconnection-id}
```

3.27.2 Request Body

None

3.27.3 Response Status Codes

```
200 OK - ExternalNetworkConnection detail
404 Not Found - ExternalNetworkConnection is not found.
```

3.27.4 Response Body

```
<externalNetworkConnection>
  <uid>{API-URL}/v1/externalnetworkconnection/enc-001</uid>
  <name>External Connection</name>
  <description>An external network</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/externalnetworkconnection/enc-001" rel="self" />
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="parent" />
  </links>
  <properties>
    <property>
      <name>external.networks</name>
      <type>uids</type>
      <uids>
        <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
      </uids>
    </property>
    <property>
      <name>assigned.zone</name>
      <type>uid</type>
      <uid>{API-URL}/v1/zone/zone-001</uid>
    </property>
    <property isReadOnly="true">
      <name>vrf</name>
      <vrf>
        <name>8a6fcd2</name>
        <devices>
          <vrfDevice>
            <ipv4>10.86.241.19</ipv4>
          </vrfDevice>
          <vrfDevice>
```

```
            <ipv4>10.86.241.20</ipv4>
          </vrfDevice>
        </devices>
      </vrf>
    </property>
    <property isReadOnly="true">
      <name>routing</name>
        <routingStrategy>
          <ospf>
            <area>1234</area>
            <devices>
              <routingDevice>
                <ipv4>10.86.241.19</ipv4>
                <process>4</process>
              </routingDevice>
              <routingDevice>
                <ipv4>10.86.241.20</ipv4>
                <process>4</process>
              </routingDevice>
            </devices>
          </ospf>
        </routingStrategy>
    </property>
  </properties>
</externalNetworkConnection>
```

3.28 Update an ExternalNetworkConnection

3.28.1 Request

PUT {API-URL}/v1/externalnetworkconnection/enc-*001* or
POST {API-URL}/v1/externalnetworkconnection/enc-*001*/put

3.28.2 Request Body

```
<externalNetworkConnection>
  <uid>{API-URL}/v1/externalnetworkconnection/enc-001</uid>
  <name>External Connection</name>
  <description>An external network 2</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/externalnetworkconnection/enc-001" rel="self" />
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="parent" />
  </links>
  <properties>
    <property>
      <name>external.networks</name>
      <type>uids</type>
      <uids>
        <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
      </uids>
    </property>
    <property>
      <name>assigned.zone</name>
      <type>uid</type>
      <uid>{API-URL}/v1/zone/zone-001</uid>
    </property>
    <property isReadOnly="true">
      <name>vrf</name>
      <vrf>
        <name>8a6fcd2</name>
        <devices>
          <vrfDevice>
            <ipv4>10.86.241.19</ipv4>
          </vrfDevice>
```

```
            <vrfDevice>
              <ipv4>10.86.241.20</ipv4>
            </vrfDevice>
          </devices>
        </vrf>
      </property>
      <property isReadOnly="true">
        <name>routing</name>
        <routingStrategy>
          <ospf>
            <area>1234</area>
            <devices>
              <routingDevice>
                <ipv4>10.86.241.19</ipv4>
                <process>4</process>
              </routingDevice>
              <routingDevice>
                <ipv4>10.86.241.20</ipv4>
                <process>4</process>
              </routingDevice>
            </devices>
          </ospf>
        </routingStrategy>
      </property>
    </properties>
</externalNetworkConnection>
```

3.28.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.28.4 Asynchronous Response Status Codes

200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – ENC was not found.
409 Conflict – Updating the ENC would cause some conflict (for instance, name already in use, version number mismatch).

3.28.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update ExternalNetworkConnection enc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.28.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update ExternalNetworkConnection enc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
```

```
</links>
<status>
  <taskStatus>success</taskStatus>
  <responseCode>200</responseCode>
</status>
<startTime>2011-05-27T11:02:34-08:00</startTime>
<endTime>2011-05-27T11:02:36-08:00</endTime>
<expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
<percentComplete>100</percentComplete>
<result xsi:type="ExternalNetworkConnectionValue">
  <uid>{API-URL}/v1/externalnetworkconnection/enc-001</uid>
  <name>External Connection</name>
  <description>An external network 2</description>
  <version>2</version>
  <links>
    <link href="{API-URL}/v1/externalnetworkconnection/enc-001" rel="self" />
    <link href="{API-URL}/v1/tenantnetworkcontainer/tnc-001" rel="parent" />
  </links>
  <properties>
    <property>
      <name>external.networks</name>
      <type>uids</type>
      <uids>
        <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
      </uids>
    </property>
    <property>
      <name>assigned.zone</name>
      <type>uid</type>
      <uid>{API-URL}/v1/zone/zone-001</uid>
    </property>
    <property isReadOnly="true">
      <name>vrf</name>
      <vrf>
        <name>8a6fcd2</name>
        <devices>
          <vrfDevice>
            <ipv4>10.86.241.19</ipv4>
          </vrfDevice>
          <vrfDevice>
            <ipv4>10.86.241.20</ipv4>
          </vrfDevice>
        </devices>
      </vrf>
    </property>
    <property isReadOnly="true">
      <name>routing</name>
      <routingStrategy>
        <ospf>
          <area>1234</area>
          <devices>
            <routingDevice>
              <ipv4>10.86.241.19</ipv4>
              <process>4</process>
            </routingDevice>
            <routingDevice>
              <ipv4>10.86.241.20</ipv4>
              <process>4</process>
            </routingDevice>
          </devices>
        </ospf>
      </routingStrategy>
    </property>
  </properties>
```

```
  </result>
</task>
```

3.28.7 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.28.8 Asynchronous Response Status Codes

```
200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – ExternalNetworkConnection was not found.
409 Conflict – For example, version number does not match.
```

3.29 Delete an ExternalNetworkConnection

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.29.1 Request

DELETE {API-URL}/v1/externalnetworkconnection/enc-*001* or
POST {API-URL}/v1/externalnetworkconnection/enc-*001*/delete

3.29.2 Request Body

None

3.29.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.29.4 Asynchronous Response Status Codes

```
200 OK – Delete succeeded
404 Not Found – ExternalNetworkConnection was not found.
```

3.29.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete ExternalNetworkConnection enc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.29.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete ExternalNetworkConnection enc-001</name>
  <version>1</version>
```

```
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.30 Create a Zone

Create a Zone entity in Network Services Manager, based on a Service Offerings in the Tenant Network Container entity catalog.

Zone Service Offerings are found in the Tenant Network Container's service catalog. (See Get Service Catalog for TNC for details.)

3.30.1 Request

```
POST {API-URL}/v1/tenantnetworkcontainer/tnc-001/catalog/create
```

3.30.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the Zone. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="zone">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-005</uid>
  <name>Internet Edge</name>
  <description>This is an Internet Edge Zone and will be reachable directly from
the Internet</description>
  <version>1</version>
  <parameters/>
</serviceOffering>
```

3.30.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.30.4 Asynchronous Response Status Codes

```
201 Created – Zone has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Service Offering was not found.
409 Conflict – Creating the Zone would cause some conflict (for instance, name
already in use).
```

3.30.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Zone zone-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
```

```
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.30.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create Zone zone-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ZoneValue">
    <uid>{API-URL}/v1/zone/zone-001</uid>
    <name>Internet Edge</name>
    <description>This is an Internet Edge Zone and will be reachable directly
from the Internet</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/zone/zone-001" rel="self" />
      <link href="{API-URL}/v1/zone/zone-001/catalog" rel="nsmr:catalog"/>
    </links>
    <properties>
      <property isReadOnly="true">
        <name>VRF</name>
        <vrf>
          <name>8a6fcd2</name>
          <devices>
            <vrfDevice>
              <ipv4>10.86.241.19</ipv4>
            </vrfDevice>
            <vrfDevice>
              <ipv4>10.86.241.20</ipv4>
            </vrfDevice>
          </devices>
        </vrf>
      </property>
    </properties>
  </result>
</task>
```

3.31 Get Zone Details

Retrieve details about a previously-created Zone entity.

3.31.1 Request

```
GET {API-URL}/v1/zone/{zone-id}
```

3.31.2 Request Body

None

3.31.3 Response Status Codes

```
200 OK – Zone detail
404 Not Found – Zone is not found.
```

3.31.4 Response Body

```
<zone>
  <uid>{API-URL}/v1/zone/zone-001</uid>
  <name>Internet Edge</name>
  <description>This is an Internet Edge Zone and will be reachable directly from
the Internet</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/zone/zone-001" rel="self" />
    <link href="{API-URL}/v1/zone/zone-001/catalog" rel="nsmr:catalog"/>
    <link href="{API-URL}/v1/networksegment/networksegment-001"/ rel="child">
  </links>
  <properties>
    <property isReadOnly="true">
      <name>VRF</name>
      <vrf>
        <name>8a6fcd2</name>
        <devices>
          <vrfDevice>
            <ipv4>10.86.241.19</ipv4>
          </vrfDevice>
          <vrfDevice>
            <ipv4>10.86.241.20</ipv4>
          </vrfDevice>
        </devices>
      </vrf>
    </property>
  </properties>
</zone>
```

NOTE

After NetworkSegments are created under a Zone, getting the Zone details will include child links to the created entities. This mechanism enables "walking" through child links and retrieving whatever elements were created via this API.

3.32 Get Service Catalog for Zone

A "Service Catalog" is a collection of "Service Catalog Entries" (or "Service Offerings") that identify services that can be created within some context. Any service offering in the catalog can then be requested.

In Network Services Manager 5.0.2, Zone Service catalogs can contain the following service offering type:

- networksegment

3.32.1 Request

```
GET {API-URL}/v1/zone/zone-001/catalog
```

This URL is the "`nsmr:catalog`" link of the Zone resource.

3.32.2 Request Body

None

3.32.3 Response Status Codes

200 OK – Catalog entries are returned.
404 Not Found – Zone was not found.

3.32.4 Response Body

```
<serviceOfferings>
  <serviceOffering type="networksegment">
    <uid>{API-URL}/v1/serviceoffering/serviceoffering-017</uid>
    <name>Layer 3 External Access VLAN</name>
    <description>A routed VLAN drawn from the pool of global external IP ad
dresses</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/serviceoffering/serviceoffering-017" rel="self"
/>
      <link href="{API-URL}/v1/zone/zone-001/catalog/create" rel="nsmr:create" /
>
    </links>
    <parameters>
      <parameter>
        <name>l3.vlan.netmask</name>
        <type>integer</type>
        <integer>24</integer>
      </parameter>
    </parameters>
  </serviceOffering>
</serviceOfferings>
```

3.33 Update a Zone

3.33.1 Request

PUT {API-URL}/v1/zone/zone-*001* or
POST {API-URL}/v1/zone/zone-*001*/put

3.33.2 Request Body

```
<zone>
  <uid>{API-URL}/v1/zone/zone-001</uid>
  <name>Internet Edge</name>
  <description>New Description</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/zone/zone-001" rel="self" />
    <link href="{API-URL}/v1/zone/zone-001/catalog" rel="nsmr:catalog"/>
    <link href="{API-URL}/v1/networksegment/networksegment-001"/ rel="child">
  </links>
  <properties>
    <property isReadOnly="true">
      <name>VRF</name>
      <vrf>
        <name>8a6fcd2</name>
        <devices>
          <vrfDevice>
            <ipv4>10.86.241.19</ipv4>
          </vrfDevice>
          <vrfDevice>
            <ipv4>10.86.241.20</ipv4>
          </vrfDevice>
        </devices>
      </vrf>
    </property>
  </properties>
</zone>
```

3.33.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.33.4 Asynchronous Response Status Codes

200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Zone was not found.
409 Conflict – Updating the Zone would cause some conflict (for instance, name already in use, version number mismatch).

3.33.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Zone zone-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.33.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update Zone zone-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ZoneValue">
    <uid>{API-URL}/v1/zone/zone-001</uid>
    <name>Internet Edge</name>
    <description>New Description</description>
    <version>2</version>
    <links>
      <link href="{API-URL}/v1/zone/zone-001" rel="self" />
      <link href="{API-URL}/v1/zone/zone-001/catalog" rel="nsmr:catalog"/>
    </links>
    <properties>
      <property isReadOnly="true">
        <name>VRF</name>
        <vrf>
          <name>8a6fcd2</name>
          <devices>
            <vrfDevice>
              <ipv4>10.86.241.19</ipv4>
            </vrfDevice>
```

```
            <vrfDevice>
              <ipv4>10.86.241.20</ipv4>
            </vrfDevice>
          </devices>
        </vrf>
      </property>
    </properties>
  </result>
</task>
```

3.33.7 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.33.8 Asynchronous Response Status Codes

```
200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – Zone was not found.
409 Conflict – For example, version number does not match.
```

3.34 Delete a Zone

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.34.1 Request

DELETE {API-URL}/v1/zone/zone-*001* or
POST {API-URL}/v1/zone/zone-*001*/delete

3.34.2 Request Body

None

3.34.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.34.4 Asynchronous Response Status Codes

```
200 OK – Delete succeeded
404 Not Found – Zone was not found.
```

3.34.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Zone zone-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.34.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete Zone zone-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.35 Create a NetworkSegment

A NetworkSegment is an abstraction that is used by Network Services Manager to represent a Layer 2 segment, such as a VLAN, together with any Network Services Manager -managed Layer 3 subnet addressing on that segment.

NetworkSegments are created by instantiating a service offering, typically found within a Zone's service catalog. (See Get Service Catalog For Zone for details.)

After it is created, each networksegment will provide a read-only property called "locations" which will contain a `segmentLocations` value. This property contains the information necessary for a northbound system to perform a VM attachment to this segment. Currently, the only supported attachment type is "vSphere". A typical locations property will look like this:

```
<property>
  <name>locations</name>
  <segmentLocations>
    <segmentLocation>
      <vSphere>
        <vc>
          <ipv4>10.10.0.1</ipv4>
          <uuid>c4718563-b3ba-41a1-957f-45790997f826</uuid>
        </vc>
        <dvs>
          <ipv4>10.11.0.3</ipv4>
          <uuid>91342b50-6c84-c72b-4045-e55ca3c41206</uuid>
        </dvs>
        <portGroup>
          <name>vlan100</name>
        </portGroup>
      </vSphere>
    </segmentLocation>
  </segmentLocations>
</property>
```

The UUIDs in the `vc` and `dvs` elements are formatted as per rfc4122 and reflect the UUIDs assigned by VMware to vCenter and the Nexus 1000 DVS, respectively.

NOTE

Although this property is of type `segmentLocations`, and can therefore include multiple `<segmentLocation>` blocks in future versions, in Network Services Manager 5.0.2, you can safely assume that it will contain only a single `<segmentLocation>` block.

3.35.1 Request

```
POST {API-URL}/v1/zone/zone-001/catalog/create
```

3.35.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the NetworkSegment. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

```
<serviceOffering type="networksegment">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-017</uid>
  <name>Layer 3 External Access VLAN</name>
  <description>A routed VLAN drawn from the pool of global external IP
addresses</description>
  <version>1</version>
  <parameters>
    <parameter>
      <name>l3.vlan.netmask</name>
      <type>integer</type>
      <integer>24</integer>
     </parameter>
  </parameters>
</serviceOffering>
```

3.35.3 Synchronous Response Status Codes

```
202 Accepted - Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.35.4 Asynchronous Response Status Codes

```
201 Created - NetworkSegment has been created. Location header gives canonical
URL.
400 Bad Request - There was an error in the input.
404 Not Found - Tenant, POD or Service Offering was not found.
409 Conflict - Creating the NetworkSegment would cause some conflict (for
instance, name already in use).
```

3.35.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create NetworkSegment nc-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.35.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create NetworkSegment networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="NetworkSegmentValue">
    <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
    <name>Layer 3 External Access VLAN</name>
    <description>A routed VLAN drawn from the pool of global external IP
addresses</description>
    <version>1</version>
    <links>
      <link href="{API-URL}/v1/zone/zone-001" rel="parent"/>
      <link href="{API-URL}/v1/networksegment/networksegment-001" rel="self" />
    </links>
    <properties>
      <property>
        <name>l3.vlan.netmask</name>
        <integer>24</integer>
      </property>
      <property isReadOnly="true">
        <name>vlan-id</name>
        <integer>1234</integer>
      </property>
      <property>
        <name>subnet</name>
        <subnet>
          <ipv4>10.1.2.0</ipv4>
          <mask>24</mask>
        </subnet>
      </property>
      <property isReadOnly="true">
        <name>gateway</name>
        <ipv4>10.1.2.3</ipv4>
      </property>
      <property isReadOnly="true">
        <name>locations</name>
        <segmentLocations>
          <segmentLocation>
            <vSphere>
              <vc>
                <ipv4>10.10.0.1</ipv4>
                <uuid>c4718563-b3ba-41a1-957f-45790997f826</uuid>
              </vc>
              <dvs>
                <ipv4>10.11.0.3</ipv4>
                <uuid>91342b50-6c84-c72b-4045-e55ca3c41206</uuid>
              </dvs>
              <portGroup>
                <name>vlan100</name>
              </portGroup>
            </vSphere>
          </segmentLocation>
```

```
        </segmentLocations>
      </property>
    </properties>
  </result>
</task>
```

3.36 Get NetworkSegment Details

Retrieve details about a previously-created networkContainer entity.

3.36.1 Request

```
GET {API-URL}/v1/networksegment/{networksegment-id}
```

3.36.2 Request Body

None

3.36.3 Response Status Codes

```
200 OK – NetworkSegment detail
404 Not Found – NetworkSegment is not found.
```

3.36.4 Response Body

```
<networkSegment>
  <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
  <name>Layer 3 External Access VLAN</name>
  <description>A routed VLAN drawn from the pool of global external IP
addresses</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/zone/zone-001" rel="parent"/>
    <link href="{API-URL}/v1/networksegment/networksegment-001" rel="self" />
    <link href="{API-URL}/v1/ipaddresspool/ipaddresspool-001"
rel="nsmr:ipaddresspool">
  </links>
  <properties>
    <property>
      <name>l3.vlan.netmask</name>
      <integer>24</integer>
    </property>
    <property isReadOnly="true">
      <name>vlan-id</name>
      <integer>1234</integer>
    </property>
    <property>
       <name>subnet</name>
       <subnet>
         <ipv4>10.1.2.0</ipv4>
         <mask>24</mask>
       </subnet>
    </property>
    <property isReadOnly="true">
      <name>gateway</name>
      <ipv4>10.1.2.3</ipv4>
    </property>
    <property isReadOnly="true">
      <name>locations</name>
      <segmentLocations>
        <segmentLocation>
          <vSphere>
            <vc>
              <ipv4>10.10.0.1</ipv4>
```

```
            <uuid>c4718563-b3ba-41a1-957f-45790997f826</uuid>
          </vc>
          <dvs>
            <ipv4>10.11.0.3</ipv4>
            <uuid>91342b50-6c84-c72b-4045-e55ca3c41206</uuid>
          </dvs>
          <portGroup>
            <name>vlan100</name>
          </portGroup>
        </vSphere>
      </segmentLocation>
    </segmentLocations>
  </property>
 </properties>
</networkSegment>
```

3.37 Update a NetworkSegment

3.37.1 Request

`PUT {API-URL}/v1/networksegment/`*`networksegment-001`* or
`POST {API-URL}/v1/networksegment/`*`networksegment-001`*`/put`

3.37.2 Request Body

```
<networkSegment>
  <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
  <name>Layer 3 External Access VLAN</name>
  <description>New Description</description>
  <version>1</version>
  <properties>
    <property>
      <name>l3.vlan.netmask</name>
      <integer>24</integer>
    </property>
    <property isReadOnly="true">
      <name>vlan-id</name>
      <integer>1234</integer>
    </property>
    <property>
      <name>subnet</name>
      <subnet>
        <ipv4>10.1.2.0</ipv4>
        <mask>24</mask>
      </subnet>
    </property>
    <property isReadOnly="true">
      <name>gateway</name>
      <ipv4>10.1.2.3</ipv4>
    </property>
    <property isReadOnly="true">
      <name>locations</name>
      <segmentLocations>
        <segmentLocation>
          <vSphere>
            <vc>
              <ipv4>10.10.0.1</ipv4>
              <uuid>c4718563-b3ba-41a1-957f-45790997f826</uuid>
            </vc>
            <dvs>
              <ipv4>10.11.0.3</ipv4>
              <uuid>91342b50-6c84-c72b-4045-e55ca3c41206</uuid>
            </dvs>
            <portGroup>
              <name>vlan100</name>
```

```
            </portGroup>
          </vSphere>
        </segmentLocation>
      </segmentLocations>
    </property>
  </properties>
</networkSegment>
```

3.37.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.37.4 Asynchronous Response Status Codes

200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – NetworkSegment was not found.
409 Conflict – Updating the NetworkSegment would cause some conflict (for instance, name already in use, version number mismatch).

3.37.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update NetworkSegment networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.37.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update NetworkSegment networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
    <link href="{API-URL}/v1/ipaddresspool/ipaddresspool-001"
rel="nsmr:ipaddresspool">
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="NetworkSegmentValue">
    <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
    <name>Layer 3 External Access VLAN</name>
    <description>New Description</description>
    <version>2</version>
    <links>
      <link href="{API-URL}/v1/zone/zone-001" rel="parent"/>
      <link href="{API-URL}/v1/networksegment/networksegment-001" rel="self" />
```

```
    </links>
    <properties>
      <property>
        <name>l3.vlan.netmask</name>
        <integer>24</integer>
      </property>
      <property isReadOnly="true">
        <name>vlan-id</name>
        <integer>1234</integer>
      </property>
      <property>
        <name>subnet</name>
        <subnet>
          <ipv4>10.1.2.0</ipv4>
          <mask>24</mask>
        </subnet>
      </property>
      <property isReadOnly="true">
        <name>gateway</name>
        <ipv4>10.1.2.3</ipv4>
      </property>
      <property isReadOnly="true">
        <name>locations</name>
        <segmentLocations>
          <segmentLocation>
            <vSphere>
              <vc>
                <ipv4>10.10.0.1</ipv4>
                <uuid>c4718563-b3ba-41a1-957f-45790997f826</uuid>
              </vc>
              <dvs>
                <ipv4>10.11.0.3</ipv4>
                <uuid>91342b50-6c84-c72b-4045-e55ca3c41206</uuid>
              </dvs>
              <portGroup>
                <name>vlan100</name>
              </portGroup>
            </vSphere>
          </segmentLocation>
        </segmentLocations>
      </property>
    </properties>
  </result>
</task>
```

3.37.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.37.8 Asynchronous Response Status Codes

200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – NetworkSegment was not found.
409 Conflict – For example, version number does not match.

3.38 Delete a NetworkSegment

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.38.1 Request

```
DELETE {API-URL}/v1/networksegment/networksegment-001 or
POST {API-URL}/v1/networksegment/networksegment-001/delete
```

3.38.2 Request Body

None

3.38.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.38.4 Asynchronous Response Status Codes

```
200 OK – Delete succeeded
404 Not Found – NetworkSegment was not found.
```

3.38.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete NetworkSegment networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.38.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete NetworkSegment networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.39 Get IPAddressPool Details

Retrieve details about a previously-created IPAddressPool entity. IPAddressPools are not created or updated explicitly through the NBI. Instead, they are created automatically during the creation of another parent resource (such as a Tenant, TNC, Zone, or NetworkSegment). Supported modifications (such as to the IP subnet ranges in a pool) are also performed by modifying parameters of the parent resource.

After it is created, the IPAddresspool is exposed as a read-only resource that can be found by following `<link rel="nsmr:ipaddresspool">` references from the parent resource.

3.39.1 Request

```
GET {API-URL}/v1/ipaddresspool/{ipaddresspool-id}
```

3.39.2 Request Body

None

3.39.3 Response Status Codes

```
200 OK - IPAddressPool detail
404 Not Found - IPAddressPool is not found.
```

3.39.4 Response Body

```
<ipAddressPool>
   <uid>{API-URL}/v1/ipaddresspool/ipaddresspool-001</uid>
   <name>SomeAddressPool</name>
   <description>test address pool</description
   <subnets>
     <subnet>
       <ipv4>192.168.1.0</ipv4>
        <mask>24</mask>
     </subnet>
     <subnet>
       <ipv4>192.168.3.0</ipv4>
        <mask>24</mask>
     </subnet>
   </subnets>
   <summary>
       <total>512</total>
       <reserved>5</reserved>
       <available>503</available>
       <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>192.168.1.1</ipv4Start>
            <ipv4End>192.168.1.254</ipv4End>
         </ipv4AddressRange>
         <ipv4AddressRange>
            <ipv4Start>192.168.3.1</ipv4End>
            <ipv4End>192.168.3.254</ipv4End>
         </ipv4AddressRange>
     </ipv4AddressRanges>
   </summary>

   <links>
       <link title="SomeAddressPool" rel="self" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/"/>
       <link title="reservations" rel="nsmr:reservations" href="{API-
URL}/v1/ipaddresspool/ipaddresspool=001/ipreservation"
       <link title="available addresses"  rel="nsmr:available" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/available"/>
       <link title="allocated" rel="nsmr:allocated" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/allocated"/>
       <link rel="nsmr:create" href="{API-URL}/v1/ipaddresspool/ipaddresspool-
001/reservation/allocate"/>
   </links>
</ipAddressPool>
```

IP Allocation Summary

The "summary" block provides an overview IP address allocation within of this address pool, including the following information:

- **total**: Number of total IPs in the pool, including network and broadcast addresses.
- **reserved**: Number of IPs covered by a reservation associated by this pool. IPs used by Network Services Manager itself or as network/broadcast addresses are not included in this count.
- **available**: Number of IPs that are available for use and thus could be reserved by the northbound system for VMs, etc. This number does take into account network and broadcast addresses as well as addresses which are not available for client use because Network Services Manager is using them itself (for instance, gateway IPs)

Reserved Versus Allocated

As shown, address pools contain two links:

```
<link title="reservations" rel="nsmr:reservations"
      href="{API-URL}/v1/ipaddresspool/ipaddresspool=001/ipreservation"
<link title="allocated" rel="nsmr:allocated"
      href="{API-URL}/v1/ipaddresspool/ipaddresspool-001/allocated"/>
```

Both contain lists of addresses that are in use, but the distinction between reservation and allocation might not be obvious.

- **Allocated** means that a range of addresses has been used internally by Network Services Manager, such as
  - Addresses used to create a subnet (network and broadcast addresses)
  - Addresses used to create a child address pool (delegation)
- **Reserved** means that a range of addresses has been set aside for use by a northbound system.

Both reserved and allocated addresses are excluded from the list of "available" IPs:

```
<link title="available addresses"  rel="nsmr:available"
      href="{API-URL}/v1/ipaddresspool/ipaddresspool-001/available"/>
```

All three links (reservations, allocated, available) return collections of reserved and allocated addresses are excluded from the list of "available" IPs.

3.40 Create an IPReservation

When a L3 network segment is created, an IPAddressPool is created and associated with that networksegment. One of the purposes of this IPAddressPool is to track which IPs are in use by Network Services Manager or the northbound system, so that neither accidentally uses an address that is already claimed by the other.

Northbound systems indicate their intent to claim exclusive use of a set of IP addresses by using a NBI resource called an IPReservation. Northbound systems can create multiple IPReservations, each containing either a single IP a set of addresses (or ranges). After the reservation is in place, the northbound system can use the addresses for VMs or any other purpose freely, and any covered addresses will not be used by Network Services Manager itself for other purposes.

The northbound system can release a reserved address by deleting the reservation object or modifying the IP addresses it covers.

IPReservations are created by using a *reservation allocator* resource that is associated with an IPAddressPool. The northbound system constructs an `<ipReservationRequest>` message and POSTs it to the allocator. The response describes the reservation that was created.

3.40.1 Request

`POST {API-URL}/v1/ipaddresspool/`*`ipaddresspool-001`*`/reservation/allocate`

3.40.2 Request Body

An ipReservationRequest can request either of the following:

- A specific number of IP addresses are to be reserved from the pool but the request does not specify the IP addresses to use. For example:

```
<ipReservationRequest>
  <ipReservation>
    <name>MyReservation</name>
    <description>test reservation</description>
    <owner>MyNorthboundSystem</owner>
  </ipReservation>
  <allocateRequest>
    <count>5</count>
  </allocateRequest>
</ipReservationRequest>
```

- Specific IP addresses are to be reserved, as a set of ranges. This request can also be used to specify a single address, as in the following example:

```
<ipReservationRequest>
   <ipReservation>
    <owner>MyNorthboundSystem</owner>
    <name>MyReservation</name>
    <description>test reservation</description>
   </ipReservation>
   <allocateRequest>
    <ipv4AddressRanges>
      <ipv4AddressRange>
         <ipv4Start>192.168.1.55</ipv4Start>
         <ipv4End>192.168.1.55</ipv4End>
      </ipv4AddressRange>
    </ipv4AddressRanges>
   </allocateRequest>
</ipReservationRequest>
```

3.40.3 Synchronous Response Status Codes

`202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.`

3.40.4 Asynchronous Response Status Codes

```
201 Created – IPReservation has been created.
400 Bad Request – There was an error in the input.
404 Not Found – Service Offering was not found.
409 Conflict – Creating the IPReservation would cause some conflict (for
instance, IP addresses were specified that are already reserved)
```

3.40.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create IPReservation within ipaddresspool-001</name>
  <version>1</version>
  <links>
```

```
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.40.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create IPReservation within ipaddresspool-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="IPReservationResponseValue">
    <ipReservation>
      <name>MyReservation</name>
      <description>test reservation</description>
      <version>1</version>
      <owner>MyNorthboundSystem</owner>
      <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>192.168.1.57</ipv4Start>
            <ipv4End>192.168.1.60</ipv4End>
         </ipv4AddressRange>
         <ipv4AddressRange>
            <ipv4Start>192.168.1.63</ipv4Start>
            <ipv4End>192.168.1.63</ipv4End>
         </ipv4AddressRange>
      </ipv4AddressRanges>
      <links>
         <link title="MyTestSubnet" rel="parent" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/"/>
         <link title="MyReservation" rel="self" href="{API-
URL}/v1/ipreservation/ipreservation-001"/>
      </links>
   </ipReservation>
   <poolSummary>
       <total>254</total>
       <reserved>7</reserved>
       <available>247</available>
       <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>192.168.1.1</ipv4Start>
            <ipv4End>192.168.1.254</ipv4End>
         </ipv4AddressRange>
       </ipv4AddressRanges>
   </poolSummary>
</result>
</task>
```

3.41 Get IPReservation Details

Retrieve details about a previously-created IPReservation entity.

3.41.1 Request

```
GET {API-URL}/v1/ipreservation/{ipreservation-id}
```

3.41.2 Request Body

None

3.41.3 Response Status Codes

```
200 OK – IPReservation detail
404 Not Found – IPReservation is not found.
```

3.41.4 Response Body

```
<ipReservation>
  <name>MyReservation</name>
  <description>test reservation</description>
  <version>1</version>
  <owner>MyNorthboundSystem</owner>
  <ipv4AddressRanges>
    <ipv4AddressRange>
      <ipv4Start>192.168.1.57</ipv4Start>
      <ipv4End>192.168.1.60</ipv4End>
    </ipv4AddressRange>
    <ipv4AddressRange>
      <ipv4Start>192.168.1.63</ipv4Start>
      <ipv4End>192.168.1.63</ipv4End>
    </ipv4AddressRange>
  </ipv4AddressRanges>
  <links>
    <link title="MyTestSubnet" rel="parent" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/"/>
    <link title="MyReservation" rel="self" href="{API-
URL}/v1/ipreservation/ipreservation-001"/>
  </links>
</ipReservation>
```

3.42 Update an IPReservation

The most common update operation is to change the `<ipv4AddressRanges>` section of the reservation, as shown in the following example.

3.42.1 Request

`PUT {API-URL}/v1/ipreservation/`*`ipreservation-001`* or
`POST {API-URL}/v1/ipreservation/`*`ipreservation-001`*`/put`

3.42.2 Request Body

```
<ipReservation>
  <name>MyReservation</name>
  <description>test reservation</description>
  <version>1</version>
  <owner>MyNorthboundSystem</owner>
  <ipv4AddressRanges>
    <ipv4AddressRange>
      <ipv4Start>192.168.1.57</ipv4Start>
      <ipv4End>192.168.1.60</ipv4End>
    </ipv4AddressRange>
    <ipv4AddressRange>
      <ipv4Start>192.168.1.63</ipv4Start>
```

```
      <ipv4End>192.168.1.65</ipv4End>
    </ipv4AddressRange>
  </ipv4AddressRanges>
  <links>
    <link title="MyTestSubnet" rel="parent" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/"/>
    <link title="MyReservation" rel="self" href="{API-
URL}/v1/ipreservation/ipreservation-001"/>
  </links>
</ipReservation>
```

3.42.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.42.4 Asynchronous Response Status Codes

200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – IPReservation was not found.
409 Conflict – Updating the IPReservation would cause some conflict (for instance, name already in use, version number mismatch).

3.42.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update IPReservation ipreservation-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.42.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update IPReservation ipreservation-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="IPReservationValue">
    <name>MyReservation</name>
    <description>test reservation</description>
    <version>2</version>
    <owner>MyNorthboundSystem</owner>
    <ipv4AddressRanges>
      <ipv4AddressRange>
```

```
        <ipv4Start>192.168.1.57</ipv4Start>
        <ipv4End>192.168.1.60</ipv4End>
      </ipv4AddressRange>
      <ipv4AddressRange>
        <ipv4Start>192.168.1.63</ipv4Start>
        <ipv4End>192.168.1.65</ipv4End>
      </ipv4AddressRange>
    </ipv4AddressRanges>
    <links>
      <link title="MyTestSubnet" rel="parent" href="{API-
URL}/v1/ipaddresspool/ipaddresspool-001/"/>
      <link title="MyReservation" rel="self" href="{API-
URL}/v1/ipreservation/ipreservation-001"/>
    </links>
  </result>
</task>
```

3.43 Delete an IPReservation

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.43.1 Request

`DELETE {API-URL}/v1/ipreservation/`*`ipreservation-001`* or
`POST {API-URL}/v1/ipreservation/`*`ipreservation-001`*`/delete`

3.43.2 Request Body

None

3.43.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.43.4 Asynchronous Response Status Codes

```
200 OK – Delete succeeded
404 Not Found – IPReservation was not found.
```

3.43.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete IPReservation ipreservation-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.43.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete IPReservation ipreservation-001</name>
```

```
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.44 ServicePolicy Overview

Network Services Manager ServicePolicies are an abstract way to define how traffic should be allowed to flow from one part of the network (within a TNC) to another.

NOTE

ServicePolicies can be defined only within a specific TNC. They cannot directly span TNC or POD boundaries. However, ServicePolicies can refer to an `ExternalNetwork` to describe traffic coming from outside that TNC.

Network Services Manager supports the following three types of ServicePolicies:

- Firewall
- NAT
- Load Balancing

In each case, the policy is configured through parameters that specify a set of network resources affected by the policy, using a flexible parameter data type called `trafficFilter`.

A `trafficFilter` type parameter is used to identify a set of network resources (ultimately, a set of IP addresses, although they can be specified using other high-level concepts) which will participate in a ServicePolicy.

Depending on the type of policy, either single or multiple `trafficFilter` parameters (source and destination, for instance) might need to be specified. This is explained in more detail in the following sections that deal with each ServicePolicy type.

TrafficFilters can identify resources in several different ways, both coarse and fine-grained. For example, a trafficFilter can specify all traffic to/from a Zone, or to/from a specific address or set of subnets within that zone.

A TrafficFilter *must* include the UID of an existing `ExternalNetwork`, `Zone`, or `NetworkSegment`, and *can* optionally refine its scope by including set of IP address ranges or addresses.

Parameters of type `trafficFilters` take a collection of multiple `<trafficFilter>` elements, and those of type `trafficFilter` take just one.

This is best demonstrated with the following examples:

- All traffic from an ExternalNetwork:

```
<parameter>
  <name>traffic.src</name>
  <trafficFilters>
    <trafficFilter>
      <uid>{API-URL}/v1/externalnetwork/864367898765</uid>
```

```
    </trafficFilter>
  </trafficFilters>
</parameter>
```

- All traffic to an Zone:

```
<parameter>
  <name>traffic.dst</name>
  <trafficFilters>
    <trafficFilter>
      <uid>{API-URL}/v1/zone/92545092649296</uid>
    </trafficFilter>
  </trafficFilters>
</parameter>
```

- All traffic to a NetworkSegment:

```
<parameter>
  <name>traffic.dst</name>
  <trafficFilters>
    <trafficFilter>
      <uid>{API-URL}/v1/networksegment/9862986776257</uid>
    </trafficFilter>
  </trafficFilters>
</parameter>
```

- Traffic from specific addresses inside an ExternalNetwork:

```
<parameter>
  <name>traffic.src</name>
  <trafficFilters>
    <trafficFilter>
      <uid>{API-URL}/v1/externalnetwork/864367898765</uid>
      <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>172.16.1.0</ipv4Start>
            <ipv4End>172.16.2.255</ipv4End>
         </ipv4AddressRange>
      </ipv4AddressRanges>
    </trafficFilter>
  </trafficFilters>
</parameter>
```

- Traffic to a single IP address inside a Zone:

```
<parameter>
  <name>vip.address</name>
  <trafficFilter>
      <uid>{API-URL}/v1/zone/92545092649296</uid>
      <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>192.168.1.99</ipv4Start>
            <ipv4End>192.168.1.99</ipv4End>
         </ipv4AddressRange>
      </ipv4AddressRanges>
  </trafficFilter>
</parameter>
```

- Traffic to multiple subnets inside a Zone:

```
<parameter>
  <name>traffic.dst</name>
  <trafficFilters>
    <trafficFilter>
      <uid>{API-URL}/v1/externalnetwork/864367898765</uid>
```

```
      <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>172.16.1.0</ipv4Start>
            <ipv4End>172.16.2.64</ipv4End>
         </ipv4AddressRange>
         <ipv4AddressRange>
            <ipv4Start>172.16.1.128</ipv4Start>
            <ipv4End>172.16.2.255</ipv4End>
         </ipv4AddressRange>
      </ipv4AddressRanges>
    </trafficFilter>
  </trafficFilters>
</parameter>
```

- Traffic from one address inside a NetworkSegment:

```
<parameter>
  <name>traffic.src</name>
  <trafficFilters>
    <trafficFilter>
      <uid>{API-URL}/v1/networksegment/9862986776257</uid>
      <ipv4AddressRanges>
         <ipv4AddressRange>
            <ipv4Start>192.168.1.99</ipv4Start>
            <ipv4End>192.168.1.99</ipv4End>
         </ipv4AddressRange>
      </ipv4AddressRanges>
    </trafficFilter>
  </trafficFilters>
</parameter>
```

3.45 Create a Firewall ServicePolicy

Firewall ServicePolicy Service Offerings are found in the Tenant's service catalog. (See Get Service Catalog For Tenant for details.) The ServiceOffering to create a Firewall ServicePolicy will be of type "`servicepolicy`" and named "`Firewall Service`".

The "`Firewall Service`" offering takes the following required parameters:

Name	Type	Description
`traffic.src`	`trafficFilters`	The source of the permitted traffic
`traffic.dst`	`trafficFilters`	The destination of the permitted traffic
`fw.service.ports`	`protocolServices`	A non-empty collection of ports and protocols describing the permitted types of traffic.

Firewall ServicePolicies implement only unidirectional traffic "permit" rules (from source to destination). Any bidirectional relationship must be established with multiple ServicePolicies.

The syntax of the `traffic.src` and `traffic.dst` parameters is described in the section ServicePolicy Overview. `fw.service.ports` is of type `protocolServices`, which represents a list of port/protocols to allow. The syntax of `protocolServices` is as follows:

- Allow all traffic:

```
<parameter>
  <name>fw.service.ports</name>
  <protocolServices>
    <protocolService>
```

```
      <protocol>any</protocol>
    </protocolService>
  </protocolServices>
</parameter>
```

- Allow a specific set of ports:

```
<parameter>
  <name>fw.service.ports</name>
  <protocolServices>
    <protocolService>
      <protocol>tcp</protocol>
      <portRange>
        <start>80</start>
        <end>97</end>
      </portRange>
    </protocolService>
    <protocolService>
      <protocol>tcp</protocol>
      <portRange>
        <start>98</start>
      </portRange>
    </protocolService>
  </protocolServices>
</parameter>
```

NOTE

If a single port is being specified, `<end>` can be omitted from the port range, as in the second `<protocolService>` block above.

3.45.1 Request

`POST {API-URL}/v1/tenant/`*`tenant-001`*`/catalog/create`

3.45.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ServicePolicy. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

The following two examples demonstrate how a Firewall ServicePolicy can be configured:

- A simple ServicePolicy, defining firewall rules between two NetworkSegments within the same tenant network container:

```
<serviceOffering type="servicepolicy">
  <uid>{API-URL}/v1/serviceoffering/offering-001</uid>
  <name>http-public-vlan-to-protected-vlan</name>
  <description>Creating FW rules between two Networks in a single
TNC</description>
  <parameters>
    <parameter>
      <name>traffic.src</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>traffic.dst</name>
      <trafficFilters>
        <trafficFilter>
```

```
          <uid>{API-URL}/v1/networksegment/networksegment-002</uid>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>fw.service.ports</name>
      <protocolServices>
        <protocolService>
          <protocol>tcp</protocol>
          <portRange>
            <start>80</start>
          </portRange>
        </protocolService>
      </protocolServices>
    </parameter>
  </parameters>
</serviceOffering>
```

- A more advanced example, creating firewall rules between a NetworkSegment and one subnet at a ExternalNetwork representing a remote site:

```
<serviceOffering type="servicepolicy">
  <uid>{API-URL}/serviceoffering/offering-001</uid>
  <name>external-network-to-protected-vlan</name>
  <description>Creating FW rules between a NetworkSegment and a remote
network</description>
  <parameters>
    <parameter>
      <name>traffic.src</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
          <ipv4AddressRanges>
             <ipv4AddressRange>
                <ipv4Start>172.16.1.0</ipv4Start>
                <ipv4End>172.16.2.255</ipv4End>
             </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>traffic.dst</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>fw.service.ports</name>
      <protocolServices>
        <protocolService>
          <protocol>tcp</protocol>
          <portRange>
            <start>80</start>
          </portRange>
        </protocolService>
      </protocolServices>
    </parameter>
  </parameters>
</serviceOffering>
```

3.45.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.45.4 Asynchronous Response Status Codes

201 Created – ServicePolicy has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ServicePolicy would cause some conflict (for instance, name already in use).

3.45.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.45.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ServicePolicyValue">
    <uid>{API-URL}/servicepolicy/servicepolicy-001</uid>
    <name>external-network-to-protected-vlan</name>
    <description>Creating FW rules between a NetworkSegment and a remote
network</description>
    <version>1</version>
    <links>
      <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-001"/>
    </links>

    <properties>
      <property>
        <name>traffic.src</name>
        <trafficFilters>
          <trafficFilter>
            <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
```

```
            <ipv4AddressRanges>
               <ipv4AddressRange>
                  <ipv4Start>172.16.1.0</ipv4Start>
                  <ipv4End>172.16.2.255</ipv4End>
               </ipv4AddressRange>
            </ipv4AddressRanges>
          </trafficFilter>
        </trafficFilters>
      </property>
      <property>
        <name>traffic.dst</name>
        <trafficFilters>
          <trafficFilter>
            <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
          </trafficFilter>
        </trafficFilters>
      </property>
      <property>
        <name>fw.service.ports</name>
        <protocolServices>
          <protocolService>
            <protocol>tcp</protocol>
            <portRange>
              <start>80</start>
            </portRange>
          </protocolService>
        </protocolServices>
      </property>
    </properties>
  </result>
</task>
```

3.46 Create a NAT ServicePolicy (Dynamic PAT)

This offering allows for masquerading via source NAT with PAT, and is found in the Tenant's service catalog. (See Get Service Catalog For Tenant for details.) The ServiceOffering to create a Dynamic PAT ServicePolicy will be of type "`servicepolicy`" and named "`Dynamic PAT Service`".

The "`Dynamic PAT Service`" offering takes the following required parameters:

Name	Type	Description
`traffic.src`	`trafficFilters`	The traffic to masquerade. Unmanaged traffic (from an ExternalNetwork) is not permitted
`traffic.dst`	`trafficFilters`	The destination of the masqueraded traffic
`mapped.address`	`trafficFilter`	A subnet of mapped addresses. An IP range must be included, and unmanaged addresses are not permitted. This would typically contain only one such address, but more are supported to cover situations in which the number of active connections exceeds the available ports for PAT.

The traffic source and mapped addresses are not permitted to be from the same zone.

The syntax of the `trafficFilters` parameters is described in the section ServicePolicy Overview.

NOTE

Using an IP address or range for `mapped.address` does not automatically mark it as "reserved". We strongly recommend that you first reserve them by Creating an IPReservation for that IP address or range.

3.46.1 Request

`POST {API-URL}/v1/tenant/tenant-001/catalog/create`

3.46.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ServicePolicy. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

In this example, dynamic PAT Masquerading is configured for all traffic from a network segment within a protected zone (*networksegment-001*) to the internet (*externalnetwork-001*), via the public zone (*zone-001*).

```
<serviceOffering type="servicepolicy">
  <uid>{API-URL}/v1/serviceoffering/offering-001</uid>
  <name>dynamic-pat-vlan</name>
  <description>masquerade protected VLAN</description>
  <parameters>
    <parameter>
      <name>traffic.src</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>traffic.dst</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>mapped.address</name>
      <trafficFilter>
        <uid>{API-URL}/v1/zone/zone-001</uid>
        <ipv4AddressRanges>
          <ipv4AddressRange>
            <ipv4Start>10.86.240.203</ipv4Start>
            <ipv4End>10.86.240.203</ipv4End>
          </ipv4AddressRange>
        </ipv4AddressRanges>
      </trafficFilter>
    </parameter>
  </parameters>
</serviceOffering>
```

3.46.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.46.4 Asynchronous Response Status Codes

201 Created – ServicePolicy has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ServicePolicy would cause some conflict (for instance, name already in use).

3.46.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-002</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.46.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-002</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ServicePolicyValue">
    <uid>{API-URL}/v1/servicepolicy/servicepolicy-002</uid>
    <name>dynamic-pat-vlan</name>
    <description>masquerade protected VLAN</description>
    <version>1</version>
    <links>
      <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-002"/>
    </links>

    <properties>
      <property>
        <name>traffic.src</name>
        <trafficFilters>
          <trafficFilter>
            <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
          </trafficFilter>
```

```
        </trafficFilters>
      </property>
      <property>
        <name>traffic.dst</name>
        <trafficFilters>
          <trafficFilter>
            <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
          </trafficFilter>
        </trafficFilters>
      </property>
      <property>
        <name>mapped.address</name>
        <trafficFilter>
          <uid>{API-URL}/v1/zone/zone-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>10.86.240.203</ipv4Start>
              <ipv4End>10.86.240.203</ipv4End>
            </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
    </property>
  </properties>
</result>
</task>
```

3.47 Create a NAT ServicePolicy (Static NAT)

This offering allows one-to-one mapping of public address to private address (all ports, bi-directional), and is found in the Tenant's service catalog. (See Get Service Catalog For Tenant for details.) The ServiceOffering to create a Dynamic PAT ServicePolicy will be of type "`servicepolicy`" and named "`Static NAT one-to-one Service`".

The "`Static NAT one-to-one Service`" offering takes the following required parameters:

Name	Type	Description
`traffic.src`	`trafficFilter`	The source traffic.
`traffic.dst`	`trafficFilter`	The destination traffic.
`mapped.address`	`trafficFilter`	The mapped address. An IP range must be included and contain only a single, managed address.

The traffic source and mapped addresses are not permitted to be from the same zone.

The syntax of these parameters is described in the section ServicePolicy Overview.

NOTE

Using an IP address as for `mapped.address` does not automatically mark it as "reserved". We strongly recommend that you first reserve it by Creating an IPReservation for that IP.

3.47.1 Request

`POST {API-URL}/v1/tenant/tenant-001/catalog/create`

3.47.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ServicePolicy. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

In this example, all traffic from the internet (*externalnetwork-001*) destined for a public IP address (*10.86.240.109*) is forwarded to a server in a protected public zone (*zone-001*) (with the IP *192.168.2.219*).

```
<serviceOffering type="servicepolicy">
  <uid>{API-URL}/v1/serviceoffering/offering-003</uid>
  <name>static-nat-to-server</name>
  <description>Redirect through DMZ</description>
  <parameters>
    <parameter>
      <name>traffic.src</name>
      <trafficFilter>
        <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
      </trafficFilter>
    </parameter>
    <parameter>
      <name>traffic.dst</name>
      <trafficFilter>
        <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        <ipv4AddressRanges>
          <ipv4AddressRange>
            <ipv4Start>192.168.2.219</ipv4Start>
            <ipv4End>192.168.2.219</ipv4End>
          </ipv4AddressRange>
        </ipv4AddressRanges>
      </trafficFilter>
    </parameter>
    <parameter>
      <name>mapped.address</name>
      <trafficFilter>
          <uid>{API-URL}/v1/zone/zone-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>10.86.240.109</ipv4Start>
              <ipv4End>10.86.240.109</ipv4End>
            </ipv4AddressRange>
          </ipv4AddressRanges>
      </trafficFilter>
    </parameter>
  </parameters>
</serviceOffering>
```

3.47.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.47.4 Asynchronous Response Status Codes

201 Created – ServicePolicy has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ServicePolicy would cause some conflict (for instance, name already in use).

3.47.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-003</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
```

```
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.47.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-003</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ServicePolicyValue">
    <uid>{API-URL}/v1/servicepolicy/servicepolicy-003</uid>
    <name>static-nat-to-server</name>
    <description>Redirect through DMZ</description>
    <version>1</version>
    <links>
      <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-003"/>
    </links>

    <properties>
      <property>
        <name>traffic.src</name>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
        </trafficFilter>
      </property>
      <property>
        <name>traffic.dst</name>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>192.168.2.219</ipv4Start>
              <ipv4End>192.168.2.219</ipv4End>
            </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </property>
      <property>
        <name>mapped.address</name>
        <trafficFilter>
          <uid>{API-URL}/v1/zone/zone-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>10.86.240.109</ipv4Start>
              <ipv4End>10.86.240.109</ipv4End>
            </ipv4AddressRange>
```

```
          </ipv4AddressRanges>
        </trafficFilter>
      </property>
    </properties>
  </result>
</task>
```

3.48 Create a NAT ServicePolicy (Static NAT with Port Redirection)

Another offering supports "DNAT" (or "port forwarding") mapping of public address/port to private address/port, and is found in the Tenant's service catalog. (See Get Service Catalog For Tenant for details.) The ServiceOffering to create a Static NAT with Port Redirection ServicePolicy will be of type "`servicepolicy`" and named "`Static NAT Service With Port Redirection`".

The "`Static NAT Service With Port Redirection`" offering takes the following required parameters:

Name	Type	Description
`traffic.src`	`trafficFilter`	The source traffic
`traffic.dst`	`trafficFilter`	The destination traffic
`nat.service.ports`	`protocolServices`	One or more of port and protocol and optional redirect port.
`mapped.address`	`trafficFilter`	The mapped address. An IP range must be included and contain only a single, managed address.

The traffic source and mapped addresses are not permitted to be from the same zone.

The syntax of these parameters is described in the section ServicePolicy Overview.

NOTE

Using an IP address as for `mapped.address` does not automatically mark it as "reserved". We strongly recommend that you first reserve it by Creating an IPReservation for that IP.

3.48.1 Request

`POST {API-URL}/v1/tenant/`*`tenant-001`*`/catalog/create`

3.48.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ServicePolicy. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

In this example, all traffic from the internet (*`externalnetwork-001`*) destined for port 29055 on a public IP address (*`10.86.240.109`*) is forwarded to port 22 on a server in a protected public zone (*`zone-001`*) (with the IP *`192.168.2.219`*).

```
<serviceOffering type="servicepolicy">
  <uid>{API-URL}/v1/serviceoffering/offering-004</uid>
  <name>static-nat-to-server</name>
  <description>SSH through DMZ</description>
  <parameters>
    <parameter>
      <name>traffic.src</name>
      <trafficFilter>
        <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
      </trafficFilter>
```

```
    </parameter>
    <parameter>
      <name>traffic.dst</name>
      <trafficFilter>
        <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        <ipv4AddressRanges>
          <ipv4AddressRange>
            <ipv4Start>192.168.2.219</ipv4Start>
            <ipv4End>192.168.2.219</ipv4End>
          </ipv4AddressRange>
        </ipv4AddressRanges>
      </trafficFilter>
    </parameter>
    <parameter>
      <name>nat.service.ports</name>
      <protocolServices>
        <protocolService>
          <protocol>tcp</protocol>
          <portRange>
            <start>29055</start>
          </portRange>
          <redirectPort>22</redirectPort>
        </protocolService>
      </protocolServices>
    </parameter>
    <parameter>
      <name>mapped.address</name>
      <trafficFilter>
          <uid>{API-URL}/v1/zone/zone-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>10.86.240.109</ipv4Start>
              <ipv4End>10.86.240.109</ipv4End>
            </ipv4AddressRange>
          </ipv4AddressRanges>
      </trafficFilter>
    </parameter>
  </parameters>
</serviceOffering>
```

3.48.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.48.4 Asynchronous Response Status Codes

201 Created – ServicePolicy has been created. Location header gives canonical URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ServicePolicy would cause some conflict (for instance, name already in use).

3.48.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-004</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
```

```
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.48.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-004</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ServicePolicyValue">
    <uid>{API-URL}/v1/servicepolicy/servicepolicy-004</uid>
    <name>static-nat-to-server</name>
    <description>SSH through DMZ</description>
    <version>1</version>
    <links>
      <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-004"/>
    </links>

    <properties>
      <property>
        <name>traffic.src</name>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
        </trafficFilter>
      </property>
      <property>
        <name>traffic.dst</name>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>192.168.2.219</ipv4Start>
              <ipv4End>192.168.2.219</ipv4End>
            </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </property>
      <property>
        <name>nat.service.ports</name>
        <protocolServices>
          <protocolService>
            <protocol>tcp</protocol>
            <portRange>
              <start>29055</start>
            </portRange>
            <redirectPort>22</redirectPort>
          </protocolService>
         <protocolServices>
      </property>
```

```
      <property>
        <name>mapped.address</name>
        <trafficFilter>
          <uid>{API-URL}/v1/zone/zone-001</uid>
          <ipv4AddressRanges>
            <ipv4AddressRange>
              <ipv4Start>10.86.240.109</ipv4Start>
              <ipv4End>10.86.240.109</ipv4End>
            </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </property>
    </properties>
  </result>
</task>
```

3.49 Create a Load Balancer ServicePolicy

Load Balancing is found in the Tenant's service catalog. (See Get Service Catalog For Tenant for details.) The ServiceOffering to create a Load Balancer ServicePolicy will be of type "`servicepolicy`" and named "`Load Balancer Service`".

The "`Load Balancer Service`" offering takes the following required parameters:

Name	Type	Description
`vip.address`	`trafficFilter`	The virtual IP address. This must be a managed address (i.e. not from an ExternalNetwork)
`lb.service.ports`	`protocolServices`	A non-empty collection of ports and protocols describing the types of traffic to balance.
`serverfarm.probe`	`strings`	A non-empty collection of probe types. Probe types supported are presented as a (multiple) `<choice>` parameter, and include `ftp`, `icmp`, `http`, `https`, `pop`, `rtsp`, `sip-tcp`, `sip-udp`, `smtp`, `radius`, `tcp`, and `udp`.
`lb.algorithm`	`string`	A string naming the algorithm to use. Load balancer algorithms supported are presented as a (single) `<choice>` parameter, and include `least-bandwidth`, `least connections`, and `round-robin`.
`lb.traffic.src`	`trafficFilters`	The source of the traffic to balance.
`real.servers`	`trafficFilters`	The back-end servers to balance to. The only permitted types of traffic are: • A `Zone` with IP ranges • A `NetworkSegment` (with or without IP ranges)

The syntax of the `trafficFilters` parameters is described in the section ServicePolicy Overview.

The real servers identified by the `real.servers` trafficFilter must all exist in the same `Zone` as `vip.address`.

A load balancer service can be updated after creation to change any of these parameters, including:

- Add/Remove real servers
- Add/Remove protocol services
- Add/Remove probes
- Change algorithm
- Change VIP address

However, the following constraints must be obeyed:

- `vip.address` trafficFilter must continue to point at the same Zone.
- `real.servers` trafficFilters must continue to point at the same (and only one) Zone.
- Any referenced networksegment UIDs in `real.servers` can be changed only if they remain in the same zone.

A Load Balancer Service created in an *Unsecured Edge Zone* does not affect security policies. Therefore, traffic originating from any source will be able to reach the VIP address of the load balancer context by default. When a the Load Balancer Service is created in a *Secured Edge Zone*, firewall rules **are** written on the Firewall Context, to allow traffic from the '`lb.traffic.src`' to the '`vip.address`'.

NOTE

Using an IP address as for `vip.address` does not automatically mark it as "reserved". We strongly recommend that you first reserve it by Creating an IPReservation for that IP.

3.49.1 Request

`POST {API-URL}/v1/tenant/`*`tenant-001`*`/catalog/create`

3.49.2 Request Body

This is the service offering with default parameter values replaced by user-selected values. Name and Description are the desired name and description of the ServicePolicy. The `<links>` section can be omitted, as shown. If it is included, it will be ignored by Network Services Manager.

In this example, all HTTP/HTTPS traffic from the internet (*`externalnetwork-001`*) destined for the public VIP (*`10.86.240.105`*) is load balanced between the private addressed servers (`192.168.2.218` and `192.168.2.221`), connected to a networksegment (*`networksegment-001`*) in a protected public zone (*`zone-001`*).

```
<serviceOffering type="servicepolicy">
  <uid>{API-URL}/v1/serviceoffering/offering-005</uid>
  <name>lb-to-web-servers</name>
  <description>HTTP through ACE</description>
  <parameters>
    <parameter>
      <name>lb.traffic.src</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
        </trafficFilter>
      </trafficFilters>
    </parameter>
    <parameter>
      <name>real.servers</name>
      <trafficFilters>
```

```
          <trafficFilter>
            <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
            <ipv4AddressRanges>
              <ipv4AddressRange>
                <ipv4Start>192.168.2.218</ipv4Start>
                <ipv4End>192.168.2.218</ipv4End>
              </ipv4AddressRange>
              <ipv4AddressRange>
                <ipv4Start>192.168.2.221</ipv4Start>
                <ipv4End>192.168.2.221</ipv4End>
              </ipv4AddressRange>
            </ipv4AddressRanges>
          </trafficFilter>
        </trafficFilters>
      </parameter>
      <parameter>
        <name>lb.service.ports</name>
        <protocolServices>
          <protocolService>
            <protocol>tcp</protocol>
            <portRange>
              <start>80</start>
            </portRange>
          </protocolService>
          <protocolService>
            <protocol>tcp</protocol>
            <portRange>
              <start>443</start>
            </portRange>
          </protocolService>
        <protocolServices>
      </parameter>
      <parameter>
        <name>vip.address</name>
        <trafficFilter>
            <uid>{API-URL}/v1/zone/zone-001</uid>
            <ipv4AddressRanges>
              <ipv4AddressRange>
                <ipv4Start>10.86.240.105</ipv4Start>
                <ipv4End>10.86.240.105</ipv4End>
              </ipv4AddressRange>
            </ipv4AddressRanges>
        </trafficFilter>
      </parameter>
      <parameter>
        <name>lb.algorithm</name>
        <string>round-robin</string>
      </parameter>
      <parameter>
        <name>serverfarm.probe</name>
        <strings>
          <string>icmp</string>
        </strings>
      </parameter>
    </parameters>
</serviceOffering>
```

3.49.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.49.4 Asynchronous Response Status Codes

```
201 Created – ServicePolicy has been created. Location header gives canonical
URL.
400 Bad Request – There was an error in the input.
404 Not Found – Tenant, POD or Service Offering was not found.
409 Conflict – Creating the ServicePolicy would cause some conflict (for
instance, name already in use).
```

3.49.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-005</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.49.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Create ServicePolicy servicepolicy-005</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>201</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ServicePolicyValue">
    <uid>{API-URL}/v1/servicepolicy/servicepolicy-005</uid>
    <name>lb-to-web-servers</name>
    <description>HTTP through ACE</description>
    <version>1</version>
    <links>
      <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-005"/>
    </links>

    <properties>
      <property>
        <name>lb.traffic.src</name>
        <trafficFilters>
          <trafficFilter>
            <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
          </trafficFilter>
        </trafficFilters>
      </property>
      <property>
        <name>real.servers</name>
        <trafficFilters>
```

```
            <trafficFilter>
              <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
              <ipv4AddressRanges>
                <ipv4AddressRange>
                  <ipv4Start>192.168.2.218</ipv4Start>
                  <ipv4End>192.168.2.218</ipv4End>
                </ipv4AddressRange>
                <ipv4AddressRange>
                  <ipv4Start>192.168.2.221</ipv4Start>
                  <ipv4End>192.168.2.221</ipv4End>
                </ipv4AddressRange>
              </ipv4AddressRanges>
            </trafficFilter>
          </trafficFilters>
        </property>
        <property>
          <name>lb.service.ports</name>
          <protocolServices>
            <protocolService>
              <protocol>tcp</protocol>
              <portRange>
                <start>80</start>
              </portRange>
            </protocolService>
            <protocolService>
              <protocol>tcp</protocol>
              <portRange>
                <start>443</start>
              </portRange>
            </protocolService>
          <protocolServices>
        </property>
        <property>
          <name>vip.address</name>
          <trafficFilter>
            <uid>{API-URL}/v1/zone/zone-001</uid>
            <ipv4AddressRanges>
              <ipv4AddressRange>
                <ipv4Start>10.86.240.105</ipv4Start>
                <ipv4End>10.86.240.105</ipv4End>
              </ipv4AddressRange>
            </ipv4AddressRanges>
          </trafficFilter>
        </property>
        <property>
          <name>lb.algorithm</name>
          <string>round-robin</string>
        </property>
        <property>
          <name>serverfarm.probe</name>
          <strings>
            <string>icmp</string>
          </strings>
        </property>
      </properties>
    </result>
  </task>
```

3.50 Get ServicePolicy Details

Retrieve details about a previously-created networkContainer entity.

3.50.1 Request

```
GET {API-URL}/v1/servicepolicy/{servicepolicy-id}
```

3.50.2 Request Body

None

3.50.3 Response Status Codes

```
200 OK – ServicePolicy detail
404 Not Found – ServicePolicy is not found.
```

3.50.4 Response Body

```
<servicePolicy>
  <uid>{API-URL}/servicepolicy/servicepolicy-001</uid>
  <name>external-network-to-protected-vlan</name>
  <description>Creating FW rules between a NetworkSegment and a remote
network</description>
  <version>1</version>
  <links>
    <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
    <link rel="self" href="{API-URL}/v1/ipreservation/servicepolicy-001"/>
  </links>

  <properties>
    <property>
      <name>traffic.src</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
          <ipv4AddressRanges>
             <ipv4AddressRange>
                <ipv4Start>172.16.1.0</ipv4Start>
                <ipv4End>172.16.2.255</ipv4End>
             </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </trafficFilters>
    </property>
    <property>
      <name>traffic.dst</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        </trafficFilter>
      </trafficFilters>
    </property>
    <property>
      <name>fw.service.ports</name>
      <protocolServices>
        <protocolService>
          <protocol>tcp</protocol>
          <portRange>
            <start>80</start>
          </portRange>
        </protocolService>
      </protocolServices>
    </property>
  </properties>
</servicePolicy>
```

3.51 Update a ServicePolicy

Any of the properties of a service policy can be updated, other than the restrictions for Load Balancer ServicePolicies described above.

3.51.1 Request

```
PUT {API-URL}/v1/servicepolicy/servicepolicy-001 or
POST {API-URL}/v1/servicepolicy/servicepolicy-001/put
```

3.51.2 Request Body

```
<servicePolicy>
  <uid>{API-URL}/servicepolicy/servicepolicy-001</uid>
  <name>external-network-to-protected-vlan</name>
  <description>Creating FW rules between a NetworkSegment and a remote
network</description>
  <version>1</version>
  <links>
    <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
    <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-001"/>
  </links>

  <properties>
    <property>
      <name>traffic.src</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
          <ipv4AddressRanges>
             <ipv4AddressRange>
                <ipv4Start>172.16.1.0</ipv4Start>
                <ipv4End>172.16.2.255</ipv4End>
             </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </trafficFilters>
    </property>
    <property>
      <name>traffic.dst</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        </trafficFilter>
      </trafficFilters>
    </property>
    <property>
      <name>fw.service.ports</name>
      <protocolServices>
        <protocolService>
          <protocol>any</protocol>
        </protocolService>
      </protocolServices>
    </property>
  </properties>
</servicePolicy>
```

3.51.3 Synchronous Response Status Codes

```
202 Accepted – Server has accepted the request. Asynchronously, the request is
executed. Response body included a Task object and Location header gives URL of
the created Task.
```

3.51.4 Asynchronous Response Status Codes

```
200 OK Update Succeeded
400 Bad Request – There was an error in the input.
404 Not Found – ServicePolicy was not found.
409 Conflict – Updating the ServicePolicy would cause some conflict (for
instance, name already in use, version number mismatch).
```

3.51.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update ServicePolicy networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.51.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Update ServicePolicy networksegment-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
    <link href="{API-URL}/v1/ipaddresspool/ipaddresspool-001"
rel="nsmr:ipaddresspool">
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
  <result xsi:type="ServicePolicyValue">
    <uid>{API-URL}/servicepolicy/servicepolicy-001</uid>
    <name>external-network-to-protected-vlan</name>
    <description>Creating FW rules between a NetworkSegment and a remote
network</description>
    <version>1</version>
    <links>
      <link rel="parent" href="{API-URL}/v1/tenant/tenant-001"/>
      <link rel="self" href="{API-URL}/v1/servicepolicy/servicepolicy-001"/>
    </links>

    <properties>
      <property>
        <name>traffic.src</name>
        <trafficFilters>
          <trafficFilter>
            <uid>{API-URL}/v1/externalnetwork/externalnetwork-001</uid>
            <ipv4AddressRanges>
              <ipv4AddressRange>
                <ipv4Start>172.16.1.0</ipv4Start>
                <ipv4End>172.16.2.255</ipv4End>
```

```
            </ipv4AddressRange>
          </ipv4AddressRanges>
        </trafficFilter>
      </trafficFilters>
    </property>
    <property>
      <name>traffic.dst</name>
      <trafficFilters>
        <trafficFilter>
          <uid>{API-URL}/v1/networksegment/networksegment-001</uid>
        </trafficFilter>
      </trafficFilters>
    </property>
    <property>
      <name>fw.service.ports</name>
      <protocolServices>
        <protocolService>
          <protocol>any</protocol>
        </protocolService>
      </protocolServices>
    </property>
  </properties>
 </result>
</task>
```

3.51.7 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.51.8 Asynchronous Response Status Codes

200 OK – Update succeeded
400 Bad Request – There was an error in the input.
404 Not Found – ServicePolicy was not found.
409 Conflict – For example, version number does not match.

3.52 Delete a ServicePolicy

The synchronous response to this request includes the task object with its URL for tracking the progress of the asynchronous de-provisioning. The provided link can be used to retrieve the task details.

3.52.1 Request

DELETE {API-URL}/v1/servicepolicy/*servicepolicy-001* or
POST {API-URL}/v1/servicepolicy/*servicepolicy-001*/delete

3.52.2 Request Body

None

3.52.3 Synchronous Response Status Codes

202 Accepted – Server has accepted the request. Asynchronously, the request is executed. Response body included a Task object and Location header gives URL of the created Task.

3.52.4 Asynchronous Response Status Codes

200 OK – Delete succeeded
404 Not Found – ServicePolicy was not found.

3.52.5 Synchronous Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete ServicePolicy servicepolicy-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>pending</taskStatus>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endtime/>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>0</percentComplete>
</task>
```

3.52.6 Get Task Synchronous Response Body (Upon successful completion)

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>Delete ServicePolicy servicepolicy-001</name>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self" />
  </links>
  <status>
    <taskStatus>success</taskStatus>
    <responseCode>200</responseCode>
  </status>
  <startTime>2011-05-27T11:02:34-08:00</startTime>
  <endTime>2011-05-27T11:02:36-08:00</endTime>
  <expiryTime>2011-05-27T11:03:34-08:00</expiryTime>
  <percentComplete>100</percentComplete>
</task>
```

3.53 Retrieve Task

When a long-running operation is invoked, in synchronous response a link to Task object is sent and the same URL can be used later to retrieve the Task object to find out the overall status of the previously invoked operation.

3.53.1 Request

```
GET {API-URL}/v1/task/task-001
```

3.53.2 Request Body

```
None
```

3.53.3 Response Status Codes

```
200 OK - Task entry is returned
404 Not Found - Task entry is not found.
```

3.53.4 Response Body

```
<task>
  <uid>{API-URL}/v1/task/task-001</uid>
  <name>token</name>
  <description>String</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/task/task-001" rel="self"/>
  </links>
  <status>
```

```
    <taskStatus>success</taskStatus>
    <logMessage>Provisioning was done successfully. </logMessage>
  </status>
  <startTime>2001-12-17T09:30:47Z</startTime>
  <lastModificationTime>2001-12-17T09:30:47Z</lastModificationTime>
  <endTime>2001-12-17T09:30:47Z</endTime>
  <expiryTime>2001-12-17T09:30:47Z</expiryTime>
  <percentComplete>100</percentComplete>
  <result>

      ------------------
  </result>
  <message>Any informative message of the operation invoked.</message>
</task>
```

3.54 Get Service Catalog Entry (ServiceOffering)

In NBI version 1.0, the serviceCatalog resources contained an abridged version of the serviceOffering resources. As a result, it was necessary to load each one individually (by following the 'self' link) to get the full details. This is no longer necessary, but it is still supported.

3.54.1 Request

```
GET {API-URL}/v1/serviceoffering/{offering-id}
```

This URL is the "self" link of the Service Offering.

3.54.2 Request Body

None

3.54.3 Response Status Codes

```
200 OK – Service Offering is returned
404 Not Found – Service Offering was not found.
```

3.54.4 Response Body

```
<serviceOffering type="tenantnetworkcontainer">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-004</uid>
  <name>Tenant Network Container</name>
  <description>The Tenant Network Container is assigned to an NSM
POD</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/serviceoffering/serviceoffering-004" rel="self" />
    <link href="{API-URL}/v1/tenant/tenant-001/catalog/create" rel="nsmr:create"
/>
  </links>
  <parameters>
    <parameter>
      <name>tnc.pod</name>
      <type>uid</type>
      <uid/>
    </parameter>
  </parameters>
</serviceOffering>
```

3.55 Get Nested Service Catalogs

NBI 1.1 adds the capability to retrieve the complete hierarchy of offerings possible below a given catalog, rather than just a single level. This shows an overview of what offerings can later be instantiated, but does not allow multiple levels to be instantiated at one time. The standard process of instantiating each level one at a time must be followed.

3.55.1 Request

```
GET {API-URL}/v1/top/catalog?deep=true
GET {API-URL}/v1/provider/{provider-id}/catalog?deep=true
GET {API-URL}/v1/tenant/{tenant-id}/catalog?deep=true
GET {API-URL}/v1/tenantnetworkcontainer/{tenantnetworkcontainer-
id}/catalog?deep=true
GET {API-URL}/v1/zone/{zone-id}/catalog?deep=true
```

3.55.2 Request Body

None

3.55.3 Response Status Codes

```
200 OK – Service Catalog is returned
404 Not Found – Service Catalog was not found.
```

3.55.4 Response Body

```
GET {API-URL}/v1/tenantnetworkcontainer/tenantnetworkcontainer-
001/catalog?deep=true

<serviceOffering type="tenantnetworkcontainer">
  <uid>{API-URL}/v1/serviceoffering/serviceoffering-004</uid>
  <name>Tenant Network Container</name>
  <description>The Tenant Network Container is assigned to an NSM
POD</description>
  <version>1</version>
  <links>
    <link href="{API-URL}/v1/serviceoffering/serviceoffering-004" rel="self" />
    <link href="{API-URL}/v1/tenant/tenant-001/catalog/create" rel="nsmr:create"
/>
  </links>
  <parameters>
    <parameter>
      <name>tnc.pod</name>
      <type>uid</type>
      <uid/>
    </parameter>
  </parameters>

  <serviceOffering type="zone">
    <name>Internet Edge</name>
    <description>This is an Internet Edge Zone and will be reachable directly
from the Internet</description>
    <parameters/>

    <serviceOffering type="externalnetworkconnection">
      <name>External Connection</name>
      <description>An external network connection describes the connection
strategy from remote locations to a zone</description>
      <parameters>
        <parameter>
          <type>uids</type>
          <name>external.networks</name>
          <uids/>
        </parameter>
      </parameters>
    </serviceOffering>

    <serviceOffering type="networksegment">
      <name>Layer 3 External Access VLAN</name>
      <description>A routed VLAN drawn from the pool of global external IP ad
dresses</description>
      <parameters>
```

```
            <parameter>
              <name>l3.vlan.netmask</name>
              <type>integer</type>
              <integer>24</integer>
            </parameter>
          </parameters>
        </serviceOffering>

      </serviceOffering>

    </serviceOffering>
```

4 Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, 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

Cisco Network Services Manager 5.0.2 API Version 1.1 Specification and Reference Guide