CHAPTER 3

Using the Network Services Manager NB API

This chapter describes how to use the Network Services Manager Northbound (NB) API. Topics include:

NB API Overview

The primary goal of the Network Services Manager NB API is to facilitate automated network provisioning in cloud environments. To aid in meeting this goal, the NB API follows REST conventions and architectural style.

The primary features of the REST architecture are that it:

- Is stateless—The server does not maintain session states for REST. Instead, the server processes each client request independently and does not rely on a session-specific context. As a result, each request issued to the server must contain all relevant information.
- Uses a client-server architecture—A uniform interface separates clients from servers, thereby freeing clients from data storage concerns and servers from a user interface or user state.
- Can cache responses—Responses are defined as cacheable or noncacheable so that clients do not use old or incorrect data for subsequent requests. By managing caching carefully, you can eliminate unneeded client-server interactions and thereby improve performance and scalability.
- Works with layered systems—Because REST interactions are stateless and can be cached, a client can interact with a server through intermediate layers, such as shared caches, load balancers, or security proxies, which can improve performance, scalability, and security.

A REST-compliant interface adheres to the following standards:

- Resource identification—In REST, a resource is a source of specific information that is identified by using a Uniform Resource Identifier (URI).
- Resource manipulation—To manipulate resources, clients and servers communicate via a standardized interface (such as HTTP) and exchange representations of these resources. A representation is in a particular encoding (such as XML), and contains some or all of the data in the resource. The data in the resource is manipulated through these representations, through the use of HTTP DELETE, PUT, or POST operations.

- Self-descriptive messages—Each message contains enough information to be parsed appropriately by the client or server.
- Hypermedia as the engine—A REST client enters a REST application through a simple fixed URL. All future actions the client takes are discovered within resource representations returned from the server. The media types used for these representations, and the link relations they contain, are standardized. The client transitions through application states by selecting from the links within a representation or by manipulating the representation in other ways afforded by its media type. In this way, the interaction is driven by hypermedia.

NB API Request Methods

The Network Services Manager NB API uses the HTTP request methods described in Table 3-1.

Table 3-1 HTTP Request Methods

HTTP Request Method	Description
GET	Retrieves the specified resource or representation. GET is a read-only operation that does not change the engine state or have any side effects.
POST	Submits data to be processed to the specified resource. The data to be processed is included in the request body. A POST operation can create a new resource, update an existing resource, or both. POST operations are defined as *nonidempotent*; that is, submitting the same POST operation more than once returns a different result for each operation. For example, if you issue a POST operation with the **create** URI to create a new tenant, a tenant is created with its own URI. If you issue the same POST operation again, another tenant is created with a different URI.
PUT	Updates the specified resource with new information. The data that is included in the PUT operation replaces the previous data. PUT operations are defined as *idempotent*; that is, submitting the same PUT operation more than once returns the same results. For example, issuing multiple PUT operations to change a tenant description from New York to Chicago returns the same result each time.
DELETE	Deletes a resource. If you delete a resource that has already been deleted, a 404 Not Found response is returned.

If you are familiar with Create, Read, Update, and Delete (CRUD) operations, the following mapping of CRUD to REST operations might help:

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

These mappings are not precise and can change over time.

NB API Responses

The NB API responds to all requests it receives with either synchronous or asynchronous responses:

- Synchronous responses—Synchronous responses are returned when little processing is required by the server. When using a REST client, synchronous responses are provided in response to GET requests because the response requires only the retrieval of information, such as a list of the network containers.
- Asynchronous responses—POST, PUT, and DELETE requests usually require more processing than a GET request, and the NB API responds by issuing a synchronous response that contains a Task object UID and a Task URL. The NB API continues to work on the initial request in the background until it is complete. The Task URL allows the NB system to determine the status of the original request as it progresses.

  The Task status object in the task response can be used by the NB system to determine the success or failure of updates to the business model. A successful update to the business model triggers the provisioning engine to run, which could also update the service directives for the controller. Provisioning and controller failures are displayed as alerts in the logs and can be viewed in the Administration UI Alert View.

The following topics provide more information about status codes and NB API responses:

- Status Codes and Error Handling, page 3-3
- Response Components, page 3-4

Status Codes and Error Handling

The NB API uses standard HTTP status codes to report the success or failure of the submitted requests:

- HTTP status codes from 200-299 indicate success.
- HTTP status codes 400 and higher indicate failure.

Table 3-2 describes common HTTP status codes and descriptions.

Table 3-2 HTTP Status Codes and Descriptions

Code	Status Reason	Description
200	OK	The request has succeeded.
201	Created	An asynchronous task has completed, and the object has been created.
202	Accepted	An asynchronous task has been accepted, but the processing is not complete.
400	Bad Request	An invalid request has been submitted. Verify that the request uses the correct syntax.
404	Not Found	The specified resource cannot be found.
409	Conflict	An attempt has been made to modify a resource that has already been modified by another request.
500	Internal Server Error	The server encountered an unexpected condition that prevents it from completing the request.

Response Components

The response might or might not contain a response body. If a response body is included:

- A successful response includes a representation of the resource.
- A failure response includes an error object.

Table 3-3 describes the components of a task response in the order in which they are given.

Table 3-3 Components of a Task Response

Information	Description
Task UID	Unique identifier for the task.
Task URL	Link to the task in the form of an HREF.
Task status	Current task status: • Pending—The task has not started processing. • In progress—The system is currently processing the task. • Success—The task completed successfully. • Failure—A problem occurred while the task was being processed.
Time stamps	Task start, end, modification, and expiration times.
Percentage task complete	Percentage of task complete.
Failure reason	Provided if the task fails.

Figure 3-1 shows an example of a task response with a failure status.

Figure 3-1 Example Failure Response

NB API Client Conventions

Adhere to the following conventions when using the Network Services Manager NB API:

- All service offering requests must include the Network Services Manager namespace information:

```
<serviceOffering xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.cisco.com/NetworkServicesManager">
```

- All request headers must include:
  - Accept: application/xml
  - Content-Type: application/xml
- Descriptions must contain no more than 250 characters.
- The NB API user must use HTTPS to invoke the northbound interface.
- All the APIs must be executed by a user with admin access to the system.