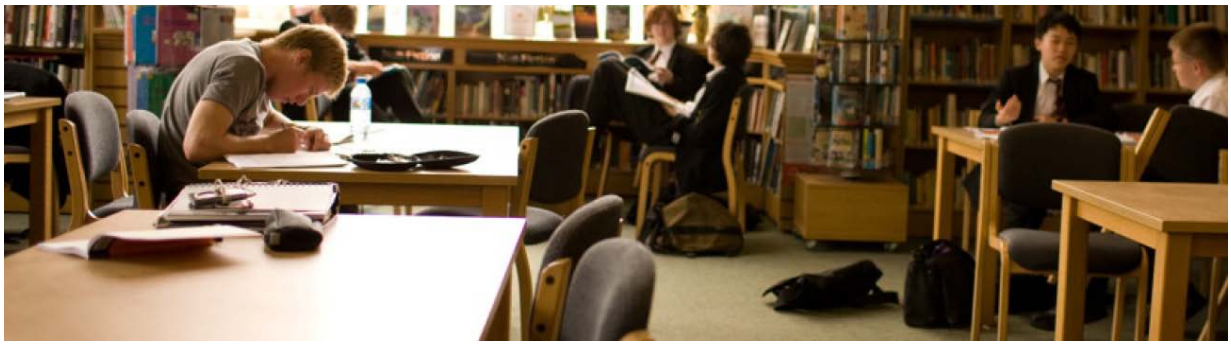
Cisco Prime Collaboration Provisioning 9.0 Northbound Interface Guide



Contents

Preface

This manual describes the Cisco Prime Collaboration Provisioning northbound interfaces. It provides instructions for using and administering it.

Conventions

This document uses the conventions as shown in Table 1

Table 1: Conventions

Item	Convention
Commands and keywords	**boldface** font
Variables for which you supply values	*italic* font
Displayed session and system information	screen font
Information you enter	**boldface** screen font
Variables you enter	The last-possible date a routine failure analysis may be performed to determine the cause of product failure or defect.
Menu items and button names	**boldface** font
Selecting a menu item in paragraphs	**Option > Network Preferences**
Selecting a menu item in tables	Option > Network Preferences

Note: Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

Chapter 1: Getting Started

This chapter describes how to start using the Cisco Prime Collaboration Provisioning Northbound Interface (Prime Collaboration Provisioning NBI) as part of the Prime Collaboration Provisioning Software Development Kit (SDK) program. It also includes feature descriptions, sample applications, typical workflow steps, and other relevant information.

This chapter contains the following sections:

- Description of the Prime Collaboration Provisioning NBI
- Feature Summary
- Functional Architecture
- Client Requirements
- Prime Collaboration Provisioning Web Services Resources
- General Prerequisites
- How to Use the Prime Collaboration Provisioning NBI SDK

Description of the Prime Collaboration Provisioning NBI

The Prime Collaboration Provisioning NBI enables external command interface. The scope of the interface is to manage basic IPT devices; manage Prime Collaboration Provisioning objects; submit work orders; and query the configured products, inventory manager, and subscriber services.

The Web Services architecture, using WSDL/SOAP over https, provides a standards-based request/response interaction with Prime Collaboration Provisioning for submitting NBI requests. Following the WS-Notification standards, Prime Collaboration Provisioning publishes a notification which is the asynchronous result of the NBI request.

Each NBI request the client sends to the server is a synchronous request message. If the Prime Collaboration Provisioning server accepts the NBI request for execution, the server returns a synchronous response. The Prime Collaboration Provisioning server executes the NBI request asynchronously. The client will receive an asynchronous notification with the NBI results when the Prime Collaboration Provisioning server has completed execution of the NBI request.

Audience

This guide is intended to be a technical resource for application developers who want to use the Prime Collaboration Provisioning NBI to provision Cisco Unified Communications deployments and implementations.

You should have an advanced level of understanding of Internet network design, operation, and terminology, and have a basic understanding of the Prime Collaboration Provisioning product.

Also, you should have an understanding of a high-level programming language such as Java, or an equivalent language. Additionally, you should have knowledge of the following:

- XML and XML Schema

- WSDL
- Web Services
- Socket programming
- Web Services standards (for example, WS-Notification, WS-Enumeration, WS-Resources, and so on).

You should be familiar with the Prime Collaboration Provisioning graphical user interface and how to use it to provision your network. In most cases, Prime Collaboration Provisioning NBI operations correlate to Prime Collaboration Provisioning operations.

Also, you should have a basic understanding of Cisco Unified Communications Manager.

Feature Summary

The Prime Collaboration Provisioning NBI provides the following:

- Management of Prime Collaboration Provisioning infrastructure (devices, Domains, and Service Areas).
- The ability to submit work orders for infrastructure products.
- The ability to query Prime Collaboration Provisioning inventory.
- The ability to submit a Prime Collaboration Provisioning work order.
- The ability to configure products.
- Management of subscriber objects.
- The ability to configure subscriber services.
- The ability to submit work orders for subscribers.
- The ability to submit get list work orders.
- One entry point for client systems to issue commands to Prime Collaboration Provisioning.
- The ability to use https or http for transport.
- The ability to retrieve list data using WS-Enumeration specification recommendations.
- The ability to query for status of any NBI request accepted by Prime Collaboration Provisioning Server.
- The listProductAttributeChoice API.

Functional Architecture

The Prime Collaboration Provisioning NBI functional architecture consists of the following:

- WSDL/XSD files with SOAP HTTP bindings exposing the NBI requests and XML-based data models for all northbound services.
- The Prime Collaboration Provisioning NBI, which receives, tracks, and manages the results of all NBI requests.
- The following Web Services are supported as part of the Prime Collaboration Provisioning NBI:

- **WS-Notification** and **WS-ResourceFramework** specifications are used to enable a client to receive the NBI request result notification asynchronously.
- **WS-Security** specifications are used to store credentials in the security token profile in the SOAP header. The AXIS2 interceptor is used to verify the credentials.
- **WS-BaseFault** specification is used to define the exceptions thrown during the synchronous submission of the NBI request. An exception is thrown if the request is not well formed or if NbiManager rejects the NBI request.
- **WS-Enumeration** specification concepts are used to retrieve data from list commands or any command capable of generating large amounts of data.

Client Requirements

The Prime Collaboration Provisioning NBI uses Web Services standards. The client must satisfy the following requirements:

- Client must be able to connect to the Prime Collaboration Provisioning server with https.
- To only receive a notification, the client must either be a Web Services Consumer or have a Web Services Consumer installed. Notifications are sent using http.

To make a request, the client does not have to be Web Services based; it can be a plain Java client.

Prime Collaboration Provisioning Web Services Resources

All services exposed by the Prime Collaboration Provisioning NBI are defined using WSDL/XSD with SOAP HTTP bindings and exposed as Web Services.

The Prime Collaboration Provisioning NBI Web Services are based on Axis 2.1.3 Web Services framework using XML Beans data binding.

The Prime Collaboration Provisioning NBI client initiates a request and response transaction to the server. If the response does not complete the request, a WS-Notification is sent to a specified server or servlet when the request is completed.

Requested bulk data is retrieved by WS-Enumeration standard requests and responses.

General Prerequisites

The Prime Collaboration Provisioning NBI SDK has been designed and tested to run on Windows XP.

It requires Java 1.6, and that the following Apache products be installed:

- Axis 2.1.3
- Tomcat 5.5.25
- Ant 1.6.5

Note: For installation information regarding these products, see the following:

- http://olex.openlogic.com/packages/axis2?show=versions&version=1645 or http://archive.apache.org/dist/ws/axis2/1_3/
- http://tomcat.apache.org/tomcat-5.5-doc/setup.html
- http://java.sun.com/javase/reference/index.jsp

If you are running the SDK on the system where Prime Collaboration Provisioning is installed, Ant will already be installed and available to you.

For CUPM clients that require an earlier version of Java, you must rebuild the CUPM NBI library. See Appendix A for details.

How to Use the Prime Collaboration Provisioning NBI SDK

The Prime Collaboration Provisioning NBI SDK provides source code, libraries, and build files for a developer to generate two Prime Collaboration Provisioning clients, and an NBI request parsing tool.

Following the information that is provided in Appendix A, “Sample Clients,” you will be able to compile and build test clients and tools, and package them in a zip file that can be installed on any platform.

The SendXmlRequest client reads XML files and sends them to the server.

The RequestSignature Java class, described in Chapter 2, “Common Prime Collaboration Provisioning NBIs,” provides methods for you to create new Prime Collaboration Provisioning clients.

Chapter 2: Common Prime Collaboration Provisioning NBIs

This chapter describes the Prime Collaboration Provisioning NBIs for operations that are common to all services.

This chapter contains the following sections:

- Prime Collaboration Provisioning NBI Overview
- Managed Objects
- Prime Collaboration Provisioning NBI Objects
- Prime Collaboration Provisioning NBI Requests

Prime Collaboration Provisioning NBI Overview

The Prime Collaboration Provisioning NBI uses the following:

- WSDL/XSD with SOAP HTTP bindings (exposed as a Web Service).
- HTTPS or HTTP as the transport layer for request/response interaction.
- HTTP for Transport Layer of notifications.
- WSDL 1.1.
- SOAP 1.2.
- AXIS2 1.3.
- XMLBeans binding framework.
- Portions of the following Web Services: WS-Notification, WS-Enumeration, WS-Security,
- WS-BaseFaults, WS-Resource, and WS-Addressing.

Table 2: XML Shemas Referenced

Prefix	Specification	URL
soap	SOAP 1.2	http://schemas.xmlsoap.org/soap/envelope/ http://schemas.xmlsoap.org/wsdl/soap/
wsdl	WSDL 1.1	http://schemas.xmlsoap.org/wsdl/
xml	—	http://www.w3.org/XML/1998/namespace
xs or xsd	XML Schema	http://www.w3.org/2001/XMLSchema
Wsrf	WS Resource Framework	http://docs.oasis-open.org/wsrf/rw-2
Wsbf	WS-BaseNotification	http://docs.oasis-open.org/wsd/bw-2
wsrf-bf	WS-BaseFaults	http://docs.oasis-open.org/wsfr/bf-2
Wsa	WS-Addressing	http://www.w3.org/2005/08/addressing
Wsen	WS-Enumeration	http://schemas.xmlsoap.org/ws/2004/09/enumeration

Wsse	WS-Security 1.1	http://docs.oasis-open.org/wss/2004/01/oasis-200401wsswssecurity-secext-1.0.xsd
	WS-ResourceProperties	http://docs.oasis-open.org/wsrf/rpw-2
wsrf-bf	WS-BaseFaults	http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd
co-v1-3	Tigerstripe	http://ossj.org.xml.Common/v1-3

Managed Objects

Fully managed objects can be created, updated, read (get), listed, and deleted by NBI requests. Partially managed objects can be read (get) and listed. Objects are hybrids of specific attributes and an array of metadata attribute/value pairs. NBI object metadata is where the attribute name is not a tag but a data field that is paired with the value (for more information on metadata, see Overview of NBI Object Metadata, page 2-35).

These pairs are defined in the schema as an array of attribute items, relating to a specific data scope. For example, the device object ArrayOfDevicePropertiesValue attribute defines names and values for the device properties. Values can be atomic, or they can be arrays of one or more unique settings. In most of the cases, valid property names are defined in enumerations and checked by the server.

The XML objects in every NBI request are validated against the XSD definition using the SAX parser.

The following Prime Collaboration Provisioning infrastructure objects are fully managed:

- Device—Network entities with IPT capabilities.
- Domain—Partitioned network for common administration.
- Service Area—Device services for subscribers.
- Subscriber—Prime Collaboration Provisioning end user.

The following are query-only objects:

- InventoryItem—A thin AXIS interface that retrieves a generic metadata object representing any object Prime Collaboration Provisioning persisted with InventoryManager. To facilitate search, associated objects are referenced by their name and attributes.
- ServiceDetail—Represents a service configured on a device. It contains products and other information, such as Domain, Service Area, subscriber information, and so on. A product object in the XSD represents a product. A service is based on either a device infrastructure configured product that supports the network configuration, or a subscriber service that supports an individual user.

The following are submit and query objects:

- Order—A work order can be submitted and its status can be tracked.
- Nbi Request—An NBI request can be retrieved and its status can be tracked. NBI requests are purged shortly after execution is completed. You can change the purge duration in the ipt.properties file.

The following are submit and query helper objects:

- CupmApiList—A generic container used to return lists of objects to the client.

- CupmApiFilter—A specific filter used for Prime Collaboration Provisioning NBI list commands. Specifies the object selection criteria and the object attribute names to return with each object.

The following are external functional objects:

- CupmApiResponseValue—The data structure returned synchronously when the client submits an NBI request that passes basic validation. Contains the NBI ID.
- CupmApiResultValue—The data structure returned asynchronously when the server completes execution of the NBI request.
- CupmApiStatus—An object in CupmApiResultValue stating the termination status of the NBI request. Also the data returned when querying an NBI request.
- CupmApiFaultType—An extension of the WS-BaseFault specification. It is populated with error information from the Prime Collaboration Provisioning server if a requests fails during the asynchronous portion of its execution.

Object Keys

Objects will be uniquely identified by their key attributes. A managed object has only one attribute as its key.

Overview of Request Names

Prime Collaboration Provisioning NBI requests will always be in the form “verb noun.”

The following list contains the verbs for basic object management:

- Create
- Get
- Update
- Delete
- List

Special verbs are used for specialized commands, such as syncDomain or resetSubscriberServicePassword. The noun for basic object management is the class.

The noun may describe an attribute or concept for specialize commands.

Overview of Object Attributes

The Prime Collaboration Provisioning NBI objects specified in the XSD files will always translate to Java POJO files. Thus there will be a Java signature corresponding to every NBI request, and a Java Prime Collaboration Provisioning NBI object for every XSD-defined object.

The structure of a Prime Collaboration Provisioning NBI object is a hybrid between specific data and metadata. All attributes requiring unique settings are defined specifically in the object. These attributes are the external key for the object.

All other attributes are in a name-value metadata structure within the object. In many classes, specifying these metadata objects is not required.

All associated objects will be referenced by their key. There will be no nesting of objects.

Overview of Request Arguments

Create<Object>

Create NBI requests will always take the managed object as an argument. Key attributes and mandatory attributes for creation must be specified in the object. All other attributes may be specified with their initial settings.

Get<Object>

Get NBI requests will always take the managed object as the first argument. The client sets only the specific attributes that define a unique key. Metadata attributes are not required or expected.

Delete<Object>

Delete NBI requests will always take the managed object as the first argument. The client sets only the specific attribute(s) that define a unique key. Metadata attributes are not required or expected.

Update<Object>

Update NBI requests will take managed object as the first argument. The user must set the attributes that are specifically defined in the object. The only metadata attributes that need be specified are those the client wants to change.

For removing a specific value from attributes that have multiple values, there is an action attribute that is a peer of the attribute to be updated. Set the action attribute to REMOVE. The values to be removed are specified by their key attributes.

List<Object>

List NBI requests take a CupmApiFilter object as the argument. The filter contains metadata pairs of attribute name and value that define the selection criteria. The metadata can be grouped to support one ‘and’ or one ‘or’ query. A null filter is allowed for generating a list of all instances of that object. The filter also has a flag indicating whether the client wants a list of object keys returned, a specified set of attributes returned, or a list of complete objects returned. Default is to return only the object keys.

Configurable Properties

The metadata configurable properties in the objects Domain, Device, Service Area, and Subscriber are defined by enumerations in the XSD files. Please refer to the html documentation provided for details.

You can find the html files in the SDK in the subdirectory html-doc folder. In the html-doc folder there are subdirectories that support the requests for that object or category. For example, for createDevice, listDevice, and so on, you go to the Device folder. Click the html file in the appropriate directory to display the documentation.

Prime Collaboration Provisioning NBI Objects

Note: Objects are traditionally supported by an interface with the naming convention <object>Value.

Device

Key attribute: deviceName

Complete Object Definition File: Device.xsd

Domain

Key attribute: domainID

Complete Object Definition File: Domain.xsd

ServiceArea

Key attributes: serviceAreaID

Complete Object Definition File: ServiceArea.xsd

InventoryItem

For querying, any object in Inventory Manager can be referenced.

Key attribute: The InventoryItem object will be set to a Prime Collaboration Provisioning IM object type. There is a metadata attribute name and value for the object type's key attributes. You must specify the IM class name and the attribute value which will uniquely identify the object in the case of a get API or a list API. Multiple instances are returned if there are no other attributes provided.

Complete Object Definition File: InventoryItem.xsd

ServiceDetail

For querying, any configured infrastructure product or any subscriber service product can be returned in a get or a list request.

Key Attribute: The key attributes for the get service requests are specified as an array of metadata in an attribute called keyAttribute. Refer to the services table in this document for specific keys for specific products. Another way to determine the keys for a product is to list all products for a device or a subscriber. The list will return only the keys and their values.

Complete Object Definition File: Order.xsd

Order

The work order ID is only used internally within the Prime Collaboration Provisioning server. Orders submitted through the user interface or batch provisioning will generate an order ID for tracking purposes.

The Prime Collaboration Provisioning client uses the NBI ID to query the work order. The getOrder API allows you to take either the order ID or the NBI ID for tracking purposes.

Complete Object Definition File: Order.xsd

Subscriber

Key attribute: subscriberID

Complete Object Definition File: Subscriber.xsd

CupmApiListValue

If the NBI request returns data, additional objects are in this container. This object is used for returning all queries handled under enumeration.

It is also used for returning prepopulation choice lists.

Complete Object Definition File: Common.xsd

CupmApiResponseValue

The ApiReponse object is the synchronously returned object for all Prime Collaboration Provisioning NBI requests that are successfully submitted to the Prime Collaboration Provisioning server.

Table 3 lists the object attributes of CupmApiResponseValue.

Table 3: CupmApiResponseValue Object Attributes

Attribute Name	Attribute Type	Description
ApiId	**String**	Unique ID within Prime Collaboration Provisioning server generated by the Prime Collaboration Provisioning server when accepting a new NBI request. Prime Collaboration Provisioning server assigns a long integer that is incremented with each new NBI request. The integer resets after 10 minutes. All NBI requests have a parameter that adds an optional string to the beginning of the generated number.
startTime	*time/date*	UTC time when server accepted NBI request.
Result	CupmApiResultValue	Not used in initial release. The framework will support synchronous execution of the Prime Collaboration Provisioning NBI request. This attribute will always be null.

Complete Object Definition File: Common.xsd

CupmApiResultValue

Table 4 lists the object attributes of CupmApiResultValue.

Table 4: CupmApiResultValue Object Attributes

Attribute Name	Attribute Type	Description
Status	**CupmApiStatusValue**	Associated CupmApiStatus object describing the final status of the NBI request as executed by Prime Collaboration Provisioning. Note that the NBI ID is in this object.
data	*CupmApiListValue*	Associated CupmApiList object containing any data the NBI request was to return.
error	CupmApiResultValue	If the NBI request fails after the response was sent, an NBI exception object will be present and describe the problem. If the NBI request executed successfully, this will be null.

Complete Object Definition File: Common.xsd

CupmApiStatusValue

Table 5 lists the object attributes of CupmApiStatusValue.

Table 5: CupmApiStatusValue Object Attributes

Attribute Name	Attribute Type	Description
apiId	**String**	NBI ID
Status	*ENUM*	Status of the NBI request
Message	String	Any informatory or warning message from a successfully executed NBI request.
startTime	time/date	UTC time when server accepted NBI request
completeTime	time/date	UTC time when server completed NBI request
OrderStatusValue	OrderStatus	An attachment (any type) for the SubmitOrder NBI

		command.

Table 6 lists the state codes for the CupmApiStatusValue Attribute.

Table 6: Status Code ENUMs for CupmApiStatusValue Attribute

State	Description
SUBMITTED	The Prime Collaboration Provisioning NBI request has been received by the Prime Collaboration Provisioning server. The synchronous message exchange is not yet completed.
REJECTED	The Prime Collaboration Provisioning NBI request was rejected by the NbiManager and an exception was thrown back to the client.
PENDING	The Prime Collaboration Provisioning NBI request has been accepted by the ApiManager and assigned an API ID. Execution is in progress.
COMPLETED_SUCCESSFULLY	The Prime Collaboration Provisioning NBI has completed successfully
COMPLETED_FAILURE	The Prime Collaboration Provisioning NBI failed. The failure could be partial or total.

For orders, an additional status object that provides information about each product detail step in the order is attached.

Complete Object Definition File: Common.xsd

CupmApiFilterValue

The CupmApiFilter has three sections:

- Object Selection Criteria for primary object—Defines which object instances to return.
    - A generic metadata structure specifying attribute names and values for selection of the data.
    - And/Or flag indicating if the above criterion is to be applied with a logical ‘and’ or a logical ‘or’.
- Object Selection Criteria for associated object—If the object type in the list selection has one or more associated objects, you can select one of these objects and apply selection criteria to this object in addition to the list object type.
    - A string containing the associated object type.
    - A generic metadata structure specifying attribute names and regular expressions for selection of the data. The and/or flag in the first selection criteria applies here as well.
- Object Attributes Returned Criteria—Specifies which attributes of the selected objects should be returned.
    - ENUM indicating return key attributes only, all attributes, or selected attributes.
    - An attribute list that is applied if the above selection requires it. The key attribute and all attributes in this list will be returned.

The CupmApiFilter class is used to define the acceptance criteria on the lists returned by the NBI list commands.

The filter contains metadata pairs of attribute name and value, where value is a string that can be set to a specific value or a regular expression. Metadata can be groups to support ‘and’ and ‘or’ queries. A null filter is allowed for generating a list of all instances of that object. For the supported string, see Infrastructure Product Key Attributes.

Table 7: Status Code ENUMS for CupmApiStatusValue Attribute

Attribute Name	Attribute Type	Description
returnedDataCriteriaValue	**ENUM**	Specifies the attributes to be returned with each object. • KEY_AND_SPECIFIED • KEYS_ONLY • ALL
additionalStatus	**anyType**	Only type ServiceDetailStatus will be the value returned by the server for the additionalStatus attribute. The ServiceDetailStatus contains the current status of each detail in a work order.

Complete Object Definition File: Common.xsd

Infrastructure Product Key Attributes

Table 8 lists the key attributes for infrastructure products. Any infrastructure product not listed does not have any key attributes.

Table 8: Infrastructure Key Attributes

Infrastructure Product	Key Attributes
Call Park	**IC_pattern**
Call Pickup Group	**IC_name**
Calling Search Space	**IC_name**
Common Device Config	**IC_name**
CTI Route Point	**IC_name**
Device Pool	**IC_name**
H323 Gateway	**IC_name**
Hunt List	**IC_name**
Hunt Pilot	**IC_pattern**
Line Group	**IC_name**
Location	**IC_name**
Media Resource Group	**IC_name**
Media Resource Group List	**IC_name**
Route Group	**IC_name**
Route List	**IC_name**
Route Partition	**IC_name**
Route Pattern	**IC_pattern/IC_routePartitionName**
SIP Trunk	**IC_name**
Translation Pattern	**IC_pattern/IC_routePartitionName**
Unified CM Group	**IC_name**
Voicemail Pilot	**IC_drin/IC_cssName**
Voicemail Profile	**IC_name**
Voice Region	**IC_name**
VG202	**IC_name**
VG204	**IC_name**
VG224	**IC_name**

Subscriber Product Key Attributes

Table 9 lists the key attributes for subscriber products. Any subscriber product not listed does not have any key attributes.

Table 9: Status Code ENUMS for CupmApiStatusValue Attribute

Subscriber Product	Key Attributes
Phone	o **type (for example, Cisco_7970)** o **macaddress (for all phone types except for Cisco_IP_Communicator, CTI_Port, CUPC)** o **devicename (only for the phone types Cisco_IP_Communicator, CTI_Port, and CUPC)**
Line	o **directorynumber** o **routepartition** o **selectedPhone**[1]
LineOnSharedPhone	o **directorynumber** o **routepartition** o **targetphone**[1]
EM_Access	**Name**
EM_Line	o **directorynumber** o **routepartition** o **SelectedEM_Access**
Voicemail	**VoicemailAlias**
Email	**Alias**
UnifiedMessaging	**SelectedEmail**

1. All selectedPhone and targetPhone names are MAC addresses. MAC addresses consist of 12 hexadecimal characters (0-9 and A-F). For a physical phone on Cisco Unified Communications Manager, the name can be the MAC address preceded by SEP (for example, SEPAAAA00001111), or the MAC address preceded by BAT in the case of a TAPS phone (for example, BATAAAA00001111). For a physical phone on Cisco Unified Communications Manager Express, nothing precedes the MAC address.

Note: For XML returned objects with keys and all product attributes, send the getSubscrberService and getConfiguredInfraProduct requests querying existing services, or the listSubscriberService and listConfiguredInfraProduct requests for subscribers and devices respectively, with a filter set to return all attributes.

Referencing Product Attributes from the Product Catalog

The product catalog defines all infrastructure and subscriber products that can be ordered using Prime Collaboration Provisioning. The product definition is in the productdirectory\schema directory in the product_catalog_translated.xsd file. Each orderable product is defined in a file in the productcatalog\metadata folder. For example, the HuntList file name is CUPMProduct_IC_HuntList.xml.

For all products, the attribute's name and type are defined in these files. Each file contains the <product> tag to define the product name and the <productAttribute> tag to define the product attribute name. The ID of the <productAttribute> tag should be used in the data file. The <name> tag of the <productAttribute> tag explains what the attribute does.

Any product provisioning attribute that is defined in the product catalog can be specified in an NBI SubmitOrder request.

For example, the SubmitOrderAddPhonesWithLines.xml file in the sample\xml directory includes the sample XML request for adding phones and lines. This XML request provisions the ID phone and the ID line.

If you want to configure the MLPP No Answer Ring Duration, refer to the CUPMProduct_line.xml file in the productcatalog\metadata directory. The provisioning attribute for this feature is mlppnard, and that it contains an integer.

To configure this feature on a line, you add the following XML code to the line productID:

```
<item>
   <attributeID>mlppnard</attributeID>
   <attributeValue>
   <co-v1-3:item>3</co-v1-3:item>
   </attributeValue>
</item>
```

This sets the MLPP No Answer Ring Duration to three rings on the line.

You can use the validate attribute to verify whether the specified setting is within the valid range. The server validates the order, but the workflow is not affected. You will receive a success notification if the setting is within the valid range; otherwise you will receive a failure message.

For an infrastructure product example, consider setting a description for a hunt list and hunt pilot.

For huntlist and huntpilot, the attribute name for description is IC_description. To add the IC_description attribute, add the following XML code to either of these product IDs:

```
<item>
   <attributeID>IC_description</attributeID>
<attributeValue>
   <co-v1-3:item>My Description</co-v1-3:item>
</attributeValue>
</item>
```

To enable Huntlist, add the following attribute. The following is from the product catalog file CUPMProduct_IC_HuntList.xml.

```
<productAttribute>
   <id>IC_routeListEnabled</id>
   <name>Enable this Hunt List</name>
```

The example of this attribute in the data file should as follows:

```
<<item>
   <attributeID>IC_routeListEnabled </attributeID>
   ------------------------------
   ------------------------------
</item>
```

> Note: All the attribute names in the product catalog are case sensitive. In most situations, even the Boolean constants are case sensitive.

Prime Collaboration Provisioning NBI Requests

All requests will synchronously return a CupmApiResponseValue object or throw an exception. Exceptions thrown are generated by WSDL/XSD or by the NbiManager.

All NBI requests will asynchronously return a CupmApiList object. The first element of the CupmApiList is an CupmApiStatus object. The second element is the requested return data, if any.

Request Signature Class

The Prime Collaboration Provisioning NBI is formally defined in the WSDL and XSD files. Each NBI request has a natural Java signature. The request can be coded as a Java class method call. The peer child objects in the request object become Java parameters to the method.

The RequestSignature class, provided in the SDK, defines this Java request signature interface. The CUPMServiceStub stub is always required as the first argument, to connect to the server. Utility commands to generate this stub are also provided in this class. This class also provides a subclass for returning data. Both the initial response document and the extracted CupmApiResultValue (when applicable) are returned.

There are some consistent changes in the Java signatures that differ from the XSD signatures. This makes the requests easier to generate in the Java environment.

Following are the consistent signature changes:

- EPR is passed in as a string. The RequestSignature class will convert the string to an EPR object containing an Address object for you.
- Enumeration values are passed in as strings.
- Provides a ResponseReturn class that returns both the original response document and the attached CupmApiResultValue.

The following sections specify the Java signature for each request, as it is defined in the RequestSignature class.

Device Management Commands and Queries

The Prime Collaboration Provisioning NBI supports full management of the Prime Collaboration Provisioning Device object.

createDevice Java Signature

Creates a new device.

Syntax:

public ResponseReturn createDevice (CUPMServiceStub stub, DeviceValue device, String endPointReference, String idPrefix);

Parameters:

- device—Device object with attribute keys and attribute name-value pairs set to their initial value.

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

CreateDeviceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus.
- CupmApiFaultType—Included if the NBI request fails.

getDevice Java Signature

Gets a device.

Syntax:

public ResponseReturn getDevice (CUPMServiceStub stub, DeviceValue device, String endPointReference, String idPrefix);

Parameters:

- device— Device object with attribute keys and attribute name-value pairs set to their initial value.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetDeviceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one Device object, the result of the get.
- CupmApiFaultType—Included if the NBI request fails.

updateDevice Java Signature

Updates a device.

Syntax:

public ResponseReturn updateDevice (CUPMServiceStub stub, DeviceValue device, String endPointReference, String idPrefix);

Parameters:

- device— Device object with attribute keys and attribute name-value pairs set to their initial value.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

UpdateDeviceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

Note: To remove a capability from a device, specify the capability in the device object and set the capabilityAction attribute to REMOVE. Note that the last capability assigned to a device cannot be removed.

listDevice Java Signature

Selects a set of devices.

Syntax:

public ResponseReturn listDevice (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- filter—Defines the object selection criteria and the device object attributes to be returned. If you only enter the version, an error is returned. A version-based query is dependent on capabilityName, so you must also enter capabilityName.

  You can filter using the following attributes:

  - DBPort
  - ExtensionMobilityServiceName

- DeviceUsername
- DoNotCreateLDAPUser
- Type
- CapabilityUsername
- CapabilityName
- CUELineUsername
- ServiceEngineInterfaceNumber
- DeviceName
- ExtensionMobilityServiceURL
- LDAPDirectoryIntegration
- CapabilityProtocol
- DeviceProtocol
- Version
- IpAddress

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListDeviceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

deleteDevice Java Signature

Deletes a device.

Syntax:

public ResponseReturn deleteDevice (CUPMServiceStub stub, DeviceValue device, String endPointReference, String idPrefix);

Parameters:

- device—Device object with key attributes set to the device to delete.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

DeleteDeviceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

getConfiguredInfraProduct Java Signature

Returns service details with only the product information, device name, and capability name. It will not return ServiceAreaID or DomainID.

Syntax:

public ResponseReturn getConfiguredInfraProduct (CUPMServiceStub stub, DeviceValue device, ProductValue product, String endPointReference, String idPrefix);

Parameters:

- device—The key attribute that is required.
- product—The infrastructure product to return. Key attributes (see Complete Object Definition File: Common.xsd, page 2-8) are used to search for the product. Nonkey attributes are ignored for product lookup.
- endPointReference—EndpointReference that the asynchronous result notification will be sent to.
- idPrefix—Precedes the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetConfiguredInfraProductException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue, containing CupmApiStatus and CupmApiList. The CupmApiList contains the one Infrastructure Product selected.
- CupmApiFaultType—Included if the NBI request fails.

listConfiguredInfraProduct Java Signature

Returns a list of all the services configured on a specific device.

Syntax:

public ResponseReturn listConfiguredInfraProduct (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- filter—Filters the returned products and determines which attributes are returned. If the client is interested in infrastructure products on only one device, the device name is specified in this filter. Device Name is the key attribute that is required.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListConfiguredInfraProductException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

Domain Management Commands and Queries

The Prime Collaboration Provisioning NBI supports full management of the Prime Collaboration Provisioning Domain object.

createDomain Java Signature

Creates a new Domain.

Syntax:

public ResponseReturn createDomain (CUPMServiceStub stub, DomainValue Domain, String endPointReference, String idPrefix);

Parameters:

- domain—Domain object with attribute keys and attribute name-value pairs set to their initial value.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

CreateDomainException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

getDomain Java Signature

Syntax:

Gets a Domain.

public ResponseReturn getDomain (CUPMServiceStub stub, DomainValue Domain, String endPointReference, String idPrefix);

Parameters:

- domain—Domain object with key attributes set to the object to retrieve.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetDomainException—If the server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one Domain object, the result of the get.
- CupmApiFaultType—Included if the NBI request fails.

updateDomain Java Signature

Updates a Domain.

Syntax:

public ResponseReturn updateDomain (CUPMServiceStub stub, DomainValue Domain, String

endPointReference, String idPrefix);

Parameters:

- Domain—Domain object with key attribute set to identify the object to update, and name-value pairs for all attributes to be updated to new values.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

UpdateDomainException—If the server cannot execute the NBI request.

ASyncReturn:

CupmApiResultValue—Contains CupmApiStatus only.

CupmApiFaultType—Included if the NBI request fails.

listDomain Java Signature

Selects a set of Domains.

Syntax:

public ResponseReturn listDomain (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- filter—Defines the object selection criteria and the device object attributes to be returned. You can filter using the following attributes:
    - SubscriberRoles
    - UnifiedMessageProcessor

- DomainID
- DomainDescription
- CallProcessor
- ServiceArea

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListDomainException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

deleteDomain Java Signature

Deletes a Domain.

Syntax:

public ResponseReturn deleteDomain (CUPMServiceStub stub, DomainValue Domain, String endPointReference, String idPrefix);

Parameters:

- domain—Domain object with key attributes set to the Domain to delete.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID

Throws:

DeleteDomainException—If the server cannot execute the NBI request.

ASyncReturn:

• CupmApiResultValue—Contains CupmApiStatus only.

• CupmApiFaultType—Included if the NBI request fails.

Service Area String Management Commands and Queries

The Prime Collaboration Provisioning NBI supports full management of the Prime Collaboration Provisioning Service Area object.

Note: The CreateServiceArea NBI allows you to associate a directory number block to the Service Area even if the Service Area is not associated to a Call Processor. The Prime Collaboration Provisioning user interface does not allow you to do this.

createServiceArea Java Signature

Creates a new Service Area.

Syntax:

public ResponseReturn createServiceArea (CUPMServiceStub stub, ServiceAreaValue serviceArea, String endPointReference, String idPrefix);

Parameters:

- serviceArea—ServiceArea object with attribute keys and attribute name-value pairs set to their initial value.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

CreateServiceAreaException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

getServiceArea Java Signature

Gets a particular Service Area.

Syntax:

public ApiResponse getServiceArea (CUPMServiceStub stub, ServiceAreaValue serviceArea, String

endPointReference, String idPrefix);

Parameters*:*

- serviceArea—ServiceArea object with key attributes set to the object to retrieve.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetServiceAreaException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one ServiceArea object, the result of the get.
- CupmApiFaultType—Included if the NBI request fails.

updateServiceArea Java Signature

Updates a Service Area.

Syntax:

public ResponseReturn updateServiceArea (CUPMServiceStub stub, ServiceAreaValue serviceArea, String endPointReference, String idPrefix);

Parameters:

- serviceArea—ServiceArea object with key attribute set to identify the object to update, and name-value pairs for all attributes to be updated to new values.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

UpdateServiceAreaException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

Note: To remove a directory number block from a Service Area, specify the directory number blocks to be removed in the Service Area property; directoryNumberBlockProperties. Then set the directoryNumberBlockAction attribute to REMOVE.

listServiceArea Java Signature

Selects a set of Service Areas.

Syntax:

public ResponseReturn listServiceArea (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- filter—Defines the object selection criteria and the device object attributes to be returned. You can filter using the following attributes:

- CommonDeviceConfig
- lastNumber
- serviceAreaID
- subscriberTemplateWithTTSDisabled
- callSearchSpaceLine
- location
- devicePool
- firstNumber
- subscriberTemplateWithTTSEnabled
- subscriberRoles
- externalMask
- cosWithTTSEnabled
- domainID
- callProcessorName
- minLength
- callSearchSpacePhone
- prefix
- cosWithTTSDisabled
- phoneProtocol
- routePartition
- unifiedMessageProcessorName
- emailProcessor

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListServiceAreaException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

deleteServiceArea Java Signature

Deletes a Service Area.

Syntax:

public ResponseReturn deleteServiceArea (CUPMServiceStub stub, ServiceAreaValue serviceArea, String endPointReference, String idPrefix);

Parameters:

- serviceArea—ServiceArea object with key attributes set to the service area to delete.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

DeleteServiceAreaException—If server cannot execute NBI request.ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

Synchronization Management Commands

Synchronization NBI requests perform the same operations as those done using the Prime Collaboration Provisioning user interface.

A synchronization status is returned, consisting of a CupmApiList with three string elements: a start time, an end time, and a message.

syncDevice Java Signature

Synchronizes the Prime Collaboration Provisioning database by polling the specified managed device. Specify a device capability to synchronize. Specify synchronization of the device infrastructure, the subscribers configured on the device, or both.

Syntax:

public ResponseReturn syncDevice (CUPMServiceStub stub, DeviceValue device, SyncType syncType, ExecuteOption executeOption, String endPointReference, String idPrefix);

Parameters:

- device—The device to synchronize; only the key attribute DeviceName is required. The Capability attribute in the Device object can be used to specify which capacities to synchronize for devices that have multiple capabilities.
- syncType—An Enum, indicating which of the two synchronizations to perform, or both. Values are INFRA, SUBSCRIBER, BOTH (default).
- executeOption—An Enum, indicating whether to perform the synchronization or whether to return the status summary of the last synchronization. Values are SYNC or LAST_SYNC_STATUS.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String added to the beginning of a numeric string generated by the server that is set to the value of the NBI ID. Null is allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

SyncDeviceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains the CupmApiStatus and an CupmApiList. The CupmApiList contains one or two CupmApiList objects, each in the synchronization status format. When two are present, the first is for Infrastructure synchronization status, and the second is for Subscriber synchronization status.
- CupmApiFaultType—Included if the NBI request fails.

syncDomain Java Signature

Performs a Prime Collaboration Provisioning synchronization of a Domain with its devices.

Syntax:

public ResponseReturn syncDomain (CUPMServiceStub stub, DomainValue Domain, ExecuteOption executeOption, String endPointReference, String idPrefix);

Parameters:

- Domain—The key attribute, DomainID is required.

- ExecuteOption—An Enum, indicating whether to perform the synchronization or whether to returnthe status summary of the last synchronization. Values are SYNC or LAST_SYNC_STATUS. The LAST_SYNC_STATUS option does not do the synchronization, but returns the status from the last synchronization.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—Precedes the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

SyncDomainException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains the CupmApiStatus and a CupmApiList. The CupmApiList contains a CupmApiList object in the synchronization status format.
- CupmApiFaultType—Included if the NBI request fails.

Note: A syncDomain command will be rejected by the server if any of the devices that belong to the Domain are running a synchronization.

Subscriber Management Commands and Queries

Prime Collaboration Provisioning NBI supports full management of the Prime Collaboration Provisioning subscriber object. It also supports some specialized commands for querying configured subscriber services and resetting passwords.

createSubscriber Java Signature

Creates one subscriber.

Syntax:

public ResponseReturn createSubscriber (CUPMServiceStub stub, SubscriberValue subscriber, String endPointReference, String idPrefix);

Parameters:

- subscriber—Subscriber object with attribute keys and attribute name-value pairs set to their initial value.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

CreateSubscriberException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

createSubscribers Java Signature

Creates multiple subscribers.

Syntax:

public ResponseReturn createSubscribers (CUPMServiceStub stub, CupmApiListValue subsriberList, String endPointReference, String idPrefix);

Parameters:

- subscriberList—CupmApiList containing a list of subscriber objects. Each subscriber object in the list is created.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

CreateSubscribersException—If server cannot execute NBI request.

ASyncReturn:

CupmApiResultValue—Contains CupmApiStatus only for successful execution.

If the result of CupmApiStatus is failure, the CupmApiList is included. The CupmApiList contains a set of CupmApiList Objects. Each CupmApiList object has an integer and a CupmApiFaultType. The integer is the 0-based index of the subscriber in the list that failed, and the CupmApiFaultType is the reason that this create subscriber failed. One CupmApiFaultType is included for the failure of the NBI request.

getSubscriber Java Signature

Gets a particular subscriber.

Syntax:

public ResponseReturn getSubscriber (CUPMServiceStub stub, SubscriberValue subscriber, String

endPointReference, String idPrefix);

Parameters:

- subscriber—Subscriber object with key attributes set to the object to retrieve.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetSubscriberException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList.
- CupmApiList—Contains one subscriber object, the result of the get.
- CupmApiFaultType—Included if the NBI request fails.

updateSubscriber Java Signature

Updates a subscriber.

Syntax:

public ApiResponse updateSubscriber (CUPMServiceStub stub, SubscriberValue subscriber, String endPointReference, String idPrefix);

Parameters:

- subscriber—Subscriber object with key attribute set to identify the object to update, and name-value pairs for all attributes to be updated to new values.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

UpdateSubscriberException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.

- CupmApiFaultType—Included if the NBI request fails.

listSubscriber Java Signature

Selects a set of subscribers.

Syntax:

public ResponseReturn listSubscriber (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- filter—Defines the object selection criteria and the device object attributes to be returned.

  You can filter using the following attributes:

  - subscriberRoles
  - lastName
  - phone
  - email
  - domainID
  - department
  - subscriberID
  - firstName
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListSubscriberException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

deleteSubscriber Java Signature

Deletes a subscriber.

Syntax:

public ResponseReturn deleteSubscriber (CUPMServiceStub stub, SubscriberValue subscriber, Boolean onlyFromCUPM, String endPointReference, String idPrefix);

Parameters:

- subscriber—Subscriber object with key attributes set to the subscriber to delete.
- onlyFromCUPM—The default is false. If set to true, the subscriber is deleted only from Prime Collaboration Provisioning. If the subscriber has any matched services in the customer record, Prime Collaboration Provisioning will disassociate (not cancel) the services and delete the user from Prime Collaboration Provisioning only. This provides a way to move a subscriber from one Domain to another or match (move) services to another Service Area.

Note: When a service is disassociated from a subscriber, the service is not deleted or disassociated on the device (processor); it is only disassociated within Prime Collaboration Provisioning. When a subsequent Domain synchronization occurs, depending on the synchronization rules, the subscriber could be created again and the services could be associated with the subscriber.

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

DeleteSubscriberException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

resetSubscriberServicePassword Java Signature

Resets the subscriber password. Provides the same function as the Manage Subscriber tab in the Prime Collaboration Provisioning user interface.

Syntax:

public ResponseReturn resetSubsciberServicePassword (CUPMServiceStub stub, SubscriberValue subscriber, String passwordType, String encryptedPword, String endPointReference, String idPrefix);

Parameters:

- subscriber—Subscriber object with the key attributes set to uniquely identify the subscriber.
- passwordType—Type of password to reset.

  Valid entries are:

  - UnifiedCMPassword
  - UnifiedCMPin
  - UnifiedCMEPassword
  - UnifiedCMDigestCredentials
  - UnityPassword
  - UnityConnectionPassword
  - UnityConnectionPin

If passwordType is null or not specified, it defaults to UnifiedCMPassword. You must set either encryptedpassword or resetToDefault, but not both.

- encryptedPassword—New password for the subscriber (encrypted).
- resetToDefault—Resets the subscriber password to the default password.
- changePasswordOnNextLogin—Forces the user to change the subscriber password on the next login.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null is allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ResetSubscriberServicePasswordException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

unlockVoiceMail Java Signature

Unlocks the voicemail PIN and password.

Syntax:

public ResponseReturn unlockVoiceMail (CUPMServiceStub stub, SubscriberValue subscriber, DeviceValue device, String unlockType, String vmalias, String notificationEpr, String idPrefix);

Parameters:

- subscriber—Subscriber object with the key attributes set to uniquely identify the subscriber.
- device—Device objects with the key attributes set to uniquely identify the subscriber.
- voicemailAlias—Voicemail alias name to be unlocked.
- unlockType—Type of the voicemail unlock, whether it is a PIN or web password or both.

  Valid entries are:

  - UnlockVoiceMailPin
  - UnlockVoiceMailWebPassword
  - UnlockVoiceMailPinWeb
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null value is allowed.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

UnlockVoiceMailException—If server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

listSubscriberService Java Signature

Returns a list of all services configured for a specific subscriber.

Syntax:

public ResponseReturn listSubscriberService (CUPMServiceStub stub, CupmApiFilterValue filter,

String endPointReference, String idPrefix);

Parameters:

- filter—Selection criteria specifies a device, Service Area, or capability. If you want services for only one subscriber, specify the subscriber ID as an attribute in the filter.

  You can filter using the following attributes:

  - SubscriberID—The key attribute that is required.
  - ProductID

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListSubscriberServiceException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

getSubscriberService Java Signature

Returns the following:

- Service details with the product information.
- Service Area ID.
- Domain ID.
- Subscriber ID.

Syntax:

public ResponseReturn getSubscriberService (CUPMServiceStub stub, SubscriberValue subscriber, ServiceAreaValue serviceArea, ProductValue product, String endPointReference, String idPrefix);

Parameters:

- Subscriber—Specify the subscriber object with the key attribute, subscriberID, to retrieve the services for.
- ServiceAreaValue—The Service Area where the subscriber's service was provisioned.
- product—Identifies the service to be returned. Key attributes (see Subscriber Product Key Attributes, page 2-10) are used to search for the product. Nonkey attributes are ignored for product lookup.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetSubscriberServiceException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

moveService Java Signature

Moves a service or a set of services of a subscriber to another Service Area.

Syntax:

public ResponseReturn moveService (CUPMServiceStub stub, SubscriberValue subscriber, ArrayOfMovingServiceDetailValue msdList, String notificationEpr, String idPrefix);

Parameters:

- Subscriber—Specify the subscriber object with the key attribute, subscriberID, to move the service.
- msdList—List of services that are to be moved.
- notificationEpr—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

MoveServiceException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

moveSubscriber Java Signature

Moves a subscriber and the selected services to a new domain.

Syntax:

public ResponseReturn moveSubscriber (CUPMServiceStub stub, SubscriberValue subscriber, String targetDomain, ArrayOfMovingServiceDetailValue msdList, String notificationEpr, String idPrefix);

Parameters:

- Subscriber—Specify the subscriber object with the key attribute, subscriberID, to retrieve the services for.
- msdList—A list of services that are to be moved.

- targetDomain—The domain to which the subscribers will be moved.
- notificationEpr—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the NBI ID returned to the client.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

MoveSubscriberException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails or contains warnings.

Overview of NBI Object Metadata

In most Prime Collaboration Provisioning NBI top-level objects, the majority of the data is metadata. For the DeviceValue, DomainValue, ServiceAreaValue, and SubscriberValue objects, only the key attributes and attributes that must be unique are specified at the top level as unique tags.

All other data is specified as metadata pair objects. The metadata object name is context-sensitive. For example, Device will have cmProperties metadata for Call Processors and umpProperties metadata for Unified Message Processors.

The metadata object is complex, and if the data always has only a single value, the object contains two strings, the propertyName and the propertyValue. Value property names are defined with an enumeration that is associated with the metadata object.

If the metadata object has one or more properties that can be assigned to multiple values, the object contains a string, an array of strings or objects, and an enumeration for action. The action ADD, which is the default, is used to add an additional value to the set of already assigned values. The action REMOVE is used to delete the specified value from the currently assigned array of values. An action assignment is only required in update requests.

Work Order Management Commands and Queries

In the Prime Collaboration Provisioning NBI, there is no concept of a prospect work order. All work orders are immediately submitted if NbiManager accepts the NBI request and if the optional validation checking is successful.

In the Prime Collaboration Provisioning NBI, there is no Work Order ID. The work orders submitted by NBI requests are identified by their NBI ID.

submitOrder Java Signature

Submits a work order to Prime Collaboration Provisioning. If the work order is accepted by Prime Collaboration Provisioning, and if it passes validation (if validation is specified), it will be submitted to Prime Collaboration Provisioning for execution.

Syntax:

public ResponseReturn submitOrder (CUPMServiceStub stub, OrderValue order, ValidationType validationType, RollbackType rollbackType, String endPointReference, String idPrefix);

Parameters:

- order—Creates a work order that is submitted to Prime Collaboration Provisioning if the NBI request is accepted. The order is never a prospect work order in Prime Collaboration Provisioning.
- validationType—ENUM as to whether a work order should be validated before being submitted. Values are VALIDATE_ONLY, SUBMIT_ONLY, VALIDATE_AND_SUBMIT.
- rollbackType—ENUM describing how the order should be rolled back if it fails. Value is PARTIAL.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String added to the beginning of the numeric value generated by the server that becomes the NBI ID. Note that in this case, the NBI ID is also used as the Order ID.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

SubmitOrderException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus only.
- CupmApiFaultType—Included if the NBI request fails.

Contents of Order Object

The order object contains the order type (ADD, CHANGE, or CANCEL), and a set of service details. These values are defined in an enumeration and they specify creating, modifying, or deleting a set of services.

The service detail object contains device name, device capability, Service Area ID, subscriber ID, Domain ID, and a set of products.

The product object contains the product ID, a sequence number, the assocWith flag, a set of key attributes, and product attributes. The sequence number defines the order of creation for each product, and it must be unique among products in the same order. The assocWith flag is used for the ADD order type. If a product that is created in one order detail needs to be associated with another product created later in the same work order, the assocWith attribute must be specified so the service being added references back to the service already added. An example of using assocWith is an order that contains multiple dependent products such as Phone, Line, Email, Voicemail, and Unified Messaging Service.

Key attributes are required in CHANGE or CANCEL order types. For key attributes, see Table and Table.

Note: The order will inherit the attribute settings of the Service Area specified in the order detail. The attributes can be specified in the order itself, overriding the Service Area settings for that one step.

Add Order

If you are sending a single order request to order multiple products (for example, Phone-Line, Phone-LineOnSharedPhone, EM_Access-EM_Line, Line-Voicemail-Email, and so on) that have a dependency, you must use sequenceNumber and assocWith.

If you are using the assocWith attribute, other attributes, such as SelectedPhone for Line product, targetphone for LineOnSharedPhone product, SelectedLine for Voicemail product, or SelectedVoicemail for Email product, must not be specified.

Cancel Order

You must specify only the key attributes of the products to be canceled. Do not specify assocWith. You must include sequenceNumber.

Change Order

The product's key attributes must be specified along with the attributes that you want to change.

- To unset a product attribute that has no subattribute, in the change order request, use the product attribute along with its empty value.

To unset a product attribute with subattributes, if you want to unset the entire provisioning attribute along with all of its subattributes, specify the parent attribute with an empty value.

If you want to unset one of the subattributes, specify only the subattributes that you want to keep in the change request.

The following example illustrates how to unset these complex provisioning attributes. This example consists of the portion of the speeddialinfo attribute with its subattributes in an Add Phone Order request:

```
<ord:attributeID>speeddialinfo</ord:attributeID>
<ord:subAttribute>
<ord:item>
<ord:attributeID>speeddialinfo_index</ord:attributeID>
<ord:attributeValue>
<v1:item>1</v1:item>
</ord:attributeValue>
</ord:item>
<ord:item>
<ord:attributeID>speeddialinfo_DirectoryNumber</ord:attributeID>
<ord:attributeValue>
<v1:item>10001</v1:item>
</ord:attributeValue>
</ord:item>
<ord:item>
<ord:attributeID>speeddialinfo_label</ord:attributeID>
<ord:attributeValue>
```

```
<v1:item>label_10001</v1:item>
</ord:attributeValue>
</ord:item>
<ord:item>
<ord:attributeID>speeddialinfo_index</ord:attributeID>
<ord:attributeValue>
<v1:item>2</v1:item>
</ord:attributeValue>
</ord:item>
<ord:item>
<ord:attributeID>speeddialinfo_DirectoryNumber</ord:attributeID>
<ord:attributeValue>
<v1:item>10002</v1:item>
</ord:attributeValue>
</ord:item>
<ord:item>
<ord:attributeID>speeddialinfo_label</ord:attributeID>
<ord:attributeValue>
<v1:item>label_10002</v1:item>
</ord:attributeValue>
</ord:item>
</ord:subAttribute>
</ord:item>
```

If you want to unset all of the speeddialinfo attribute subattributes, the change order request looks like the following:

```
<ord:attributeID>speeddialinfo</ord:attributeID>
<ord:attributeValue></ord:attributeValue>
```

If you want to unset the speeddialinfo attribute where the index value is 2, the change request looks like the following:

```
<item>
<attributeID>speeddialinfo</attributeID>
<subAttribute>
<item>
<attributeID>speeddialinfo_DirectoryNumber</attributeID>
<attributeValue>
<v1:item>10011</v1:item>
</attributeValue>
</item>
<item>
<attributeID>speeddialinfo_label</attributeID>
<attributeValue>
<v1:item>label_10001</v1:item>
```

```
</attributeValue>
</item>
<item>
<attributeID>speeddialinfo_index</attributeID>
<attributeValue>
<v1:item>1</v1:item>
</attributeValue>
</item>
</subAttribute>
</item>
```

getOrder Java Signature

Returns an order object that was previously submitted.

Syntax:

public ResponseReturn getOrder (CUPMServiceStub stub, String nbiId, String orderId, String endPointReference, String ipPre);

Parameters:

- nbiId—The key for specifying the work order object. This can retrieve only orders submitted by an NBI SubmitOrder request.
- orderId—Used to retrieve an order generated by the GUI or an NBI request. Note that nbiID or OrderID is required, but it is an error to include both.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- ipPre—Precedes the NBI ID to be assigned to this get NBI request.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetOrderException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and a CupmAPiList. The CupmApiList contains one object, the order selected. If the NBI order request is still retained, it will be returned. If its retention time is exceeded and it has been purged, a translation of the XMLEntity version of the work order object is returned.
- CupmApiFaultType—Included if the NBI request fails.

listOrder Java Signature

Returns all order details.

When using this API, remember the following:

- This is a CPU intensive operation. It will impact performance. Keep the performance limitations in mind when using this API.

Note: Using the ALL return criterion consumes a large portion of your CPU resources. It is recommended that you use the KEYS_ONLY filter option to get the list of order IDs, and then do a getOrder to get details of a specific order.

- If there are orders in the system from a previous version of Prime Collaboration Provisioning, not all the information will be returned, or the entire order may be skipped.
- If the device's name is changed after the order is placed, the IM look up may fail. If this occurs, the order is skipped.

Syntax:

public ResponseReturn listOrder (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String ipPre);

Parameters:

- filter—Used to further narrow the selection criteria.

You can filter using the following attributes:

  - SubscriberID
  - ServiceAreaID
  - DomainID
  - ProductID
  - OrderStatus

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListOrderException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards. If the NBI order request is still retained, it will be returned. If its retention time is exceeded and it has been purged, a translation of the XMLEntity version of the work order object is returned. If a filter that is set to KEYS_ONLY is provided, the CupmApiList will contain xmlstrings. Each string will contain one Object ID.
- CupmApiFaultType—Included if the NBI request fails.

Note: The pullOp API can be used to retrieve a list of elements according to a specified enumeration context (for more information on pullOp, see pullOp Java Signature.

Check Connectivity Request

ping Java Signature

Syntax:

public ResponseReturn ping (CUPMServiceStub stub, String epr)

Returns:

Two strings, the Prime Collaboration Provisioning version and the Prime Collaboration Provisioning NBI version.

AsyncReturn:

A test string containing server status is sent to the specified notification consumer at the EPR provided.

Inventory Queries

getInventoryItem Java Signature

Retrieves an inventory item. The getInventoryItem request requires only the attributes necessary to select a unique instance. Including nonkey attributes in this request is not an error, but it is not recommended as nonmatching values or incorrect attribute names will cause a failure.

Syntax:

public ResponseReturn getInventoryItem (CUPMServiceStub stub, InventoryItemValue item, String endPointReference, String idPrefix);

Parameters:

- item—An InventoryItem object that specifies the object type to retrieve, and the attribute keys and unique values for those keys.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetInventoryItemException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. The CupmApiList contains the one InventoryItem object selected.
- CupmApiFaultType—Included if the NBI request fails.

listInventoryItem Java Signature

Retrieves a list of inventory items from the Prime Collaboration Provisioning database, based on the filter selection. Note that the filter can also select based on associated objects (one level of association only).

Syntax:

public ResponseReturn listInventoryItem (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- filter—Specifies which InventoryItem objects to return and which attributes to include.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListInventoryItemException—If the server cannot execute the NBI request.

ASyncReturn:

CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.

CupmApiFaultType—Included if the NBI request fails.

Prepopulation

Prepopulation in this context refers to populating a service attribute choice list dynamically.

listProductAttributeChoice Java Signature

Returns a choice list for a specific attribute for a specific product. The choice list is dynamic, based on the Capabilityname, DeviceName, ServiceAreaID, and product attributes that have already been specified.

Syntax:

public ResponseReturn listProductAttributeChoice (CUPMServiceStub stub, String[] arrayOfAttributes, ProductValue product, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

- product—The product specified for the return choice list, containing all attributes that have already been set.
- arrayOfAttributes—Each attribute for which a choice list will be returned.
- filter—The selection criteria section of the filter specifies the device, the capability of the device, the ServiceArea, and any product attribute values that have been set. You must specify capabilityName and either DeviceName or ServiceAreaID or both.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListProductAttributeChoiceException—If the server cannot execute the NBI request.

ASyncReturn:

- CupmApiResultValue, containing CupmApiStatus and CupmApiList. CupmApiList contains the choice lists requested for the product attributes.
- CupmApiFaultType—Included if the NBI request fails.

Prime Collaboration Provisioning NBI Status Queries

getApiStatus Java Signature

Returns the status of a specific NBI request. If the execution of the command has been completed, an attached CupmApiResultValue is also returned.

Syntax:

public ResponseReturn getApiStatus (CUPMServiceStub stub, String apiId, String endPointReference, String idPrefix);

Parameters:

- apiId—The NBI ID of the command status to return.

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

GetApiStatusExcepton—If the server cannot execute the NBI request.

SyncReturn:

- CupmApiResponseValue—Contains CupmApiResultValue, which contains CupmApiStatus and CupmApiList. The list contains one object, the CupmApiStatus for the NBI request specified in the request.
- CupmApiFaultType—Included if the NBI request fails.

AsyncReturn:

None—The command is synchronous.

Note: The command will fail if the server has purged the NBI request.

listApiStatus Java Signature

Returns the status of all outstanding NBI requests this client has submitted.

Syntax:

public ResponseReturn listApiStatus (CUPMServiceStub stub, CupmApiFilterValue filter, String endPointReference, String idPrefix);

Parameters:

filter—Limited filtering of the NBI requests returned.

You can filter using the following attribute:

- status
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.

- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListApiStatusExcepton—If the server cannot execute the NBI request.

SyncReturn:

- CupmApiResponseValue—Contains CupmApiResultValue which contains CupmApiStatus andCupmApiList. CupmApiList contains one object, the EnumerationContextType, which is used to retrieve the list using WS-Enumeration suggested standards.
- CupmApiFaultType—Included if the NBI request failed or had warnings.

Async return:

None—The command is synchronousSpecialized Requests

listSrstPhoneInfo Java Signature

Generates a list of PhoneInfo objects (each represents a Phone with its Users and Lines) that are associated with the given SRST IP address and Prime Collaboration Provisioning IP address. The list of phones generated will always be the set or a subset of the phones in the Prime Collaboration Provisioning IM database.

Syntax:

public ResponseReturn listSrstPhoneInfo (CUPMServiceStub stub, String cpIpAddr, String srstIpAddr, String queryOption, String endPointReference, String idPrefix);

Paramaters:

- cpIpAddr—IP address of the Call Processor.
- srstIpAddr—IP address of the SRST.
- queryOption—The data selection criteria, defined by the PhoneInfoqueryOption enumeration.
    - ALL—(Default) all phones in Prime Collaboration Provisioning will be considered.
    - IN_CUPM_SUBSCRIBER_RECORD—Only the phones that are in the Prime Collaboration Provisioning subscriber record will be selected.

- NOT_IN_CUPM_SUBSCRIBER_RECORD—Only the phones that exist in Prime Collaboration Provisioning and do not exist in the Prime Collaboration Provisioning subscriber record are selected.

- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. A null or empty string is valid for no changes to the internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

ListSrstPhoneInfoException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains the WS-Enumeration context object, which is used as the key to pull the list.
- CupmApiFaultType—Included if the NBI request fails.

getOrderSummary Java Signature

Syntax:

public ResponseReturn getOrderSummmary (CUPMServiceStub stub, Calendar startTime, Calendar endTime, String endPointReference, String idPrefix);

Parameters:

- startTime—The beginning of the time period to report on. If not specified, the value defaults to the earliest known work order.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. A null or empty string is valid for no changes to the internal generation.

Returns:

- CupmApiResponseValue—Object with NBI ID.

Throws:

- GetOrderSummaryException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object, the OrderSummary object
- CupmApiFaultType—Included if the NBI request fails.

submitConfigTemplate Java Signature

Sends a set of Cisco IOS telephony commands, any router-related commands, and infrastructure product ordering commands to a capability on a device. The commands are specified in a template. Keyword substitution functionality is available.

Syntax:

public ResponseReturn submitConfigTemplate (CUPMServiceStub stub, ConfigDetailValue configDetail, String validate, String rollback, String endPointReference, String idPrefix);

Parameters:

- configDetail—This object defines the device and capability to send the Cisco IOS commands to. It specifies the template to send, and it contains a list of the keywords and their value for this usage.
- validate—VALIDATE, SUBMIT, or VALIDATE_AND_SUBMIT are the settings for this enumeration.
- rollback—PARTIAL_ROLLBACK setting.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

submitConfigTemplateException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains status of the execution of the Cisco IOS commands on the device.
- CupmApiFaultType—Included if the NBI request fails.

listConfigTemplateKeyword Java Signature

Provides support for listing the keywords used in templates submitted by the submitConfigTemplate request. Returns a list of all the keywords in the specified template.

Syntax:

public ResponseReturn listConfigTemplateKeyword (CUPMServiceStub stub String templateName, String endPointReference, String idPrefix);

Parameters:

- templateName—Name identifying the template for which to return the list of keywords.
- endPointReference—String to convert to an EndpointReference object that the asynchronous result notification will be sent to.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the NBI ID. Null or empty string valid for no change to internal generation.

Returns:

CupmApiResponseValue—Object with NBI ID.

Throws:

listConfigTemplateException—If server cannot execute NBI request.

ASyncReturn:

- CupmApiResultValue—Contains CupmApiStatus and CupmApiList. CupmApiList contains one object that lists the keywords in this template. That object is a KeywordListDocument containing an ArrayOfString object.
- CupmApiFaultType—Included if the NBI request fails.

WS-Enumeration Requests

When a Prime Collaboration Provisioning NBI List Command is executed successfully, a CupmApiResultValue is sent as a notification to the client. This notification contains the object wsen:EnumerationContext. The client can now send the following requests to the server while referencing this EnumerationContext:

- PullOp—Returns a list of elements according to a specified enumeration context. It may also return an updated enumeration context, which the caller should use for all future requests.
- RenewOp—Renews the expiration time for a specified enumeration context as long as the context is still valid. This operation will fail and generate a fault if the context is invalid.
- GetAPIStatus—Returns the status of an enumeration context, such as the expiration time.

- ReleaseOp—Invalidates a specified enumeration context and releases any resources allocated to theenumeration. The UUID list generated by the Prime Collaboration Provisioning NBI List command is flagged for deletion.

Note: The objects used in the Java code can be imported from org.xmlsoap.schemas.ws.x2005.x09.enumeration.

pullOp Java Signature

The pullOp command retrieves a portion of the list data generated by any list NBI. The client may set the MaxElements attribute in the CupmApiEnumIterator before sending to specify the number of elements returned. The default is 1. The pullOp command will never return more than this amount. It may return less if some objects have been deleted since the report was generated, or if there was an error. Subsequent calls to pull will return the next sequence of elements in the data.

Syntax:

public ResponseReturn pullOp (CUPMServiceStub stub, wsen:EnumerationContext, wsen:MaxElements, pullSize);

Parameters:

- wsen:EnumerationContext—The latest version of this object that the client has received from the server.
- wsen:MaxElements, pullSize—The number of objects to return with this request.

Returns:

Standard Pull response.

Throws:

- RemoteExcepton—If the server cannot execute the NBI request.

SyncReturn:

The standard pull response, which contains an ItemArrayType. In that type is the number of objects requested. If the pull reached the end of the list, a wsen:EndOfSequence object is returned as a peer of the list. This indicates to the client that no more elements are available and that the enumeration context is now invalid. The client should not invoke additional pull operations or invoke the Release operation to signal that the enumeration context is no longer needed. Doing so will result in a wsen:InvalidEnumerationContext fault.

CupmApiFaultType—Included if the NBI request fails.

renewOp Java Signature

The renewOp command tells the server that the client wants to keep the data longer than the initial expiration date. The Prime Collaboration Provisioning server will allow this to happen only if the throttling thresholds for reports will not be exceeded by the renewal.

Syntax:

public ResponseReturn renewOp (CUPMServiceStub stub, wsen:EnumerationContext enumerationContext, XmlCalendar Date or Duration duration);

If a date is specified, that date becomes the new expiration date. If a duration is specified, that time is added to the current expiration date.

The NBI properties control the maximum number of times an expiration date can be increased.

Parameters:

- wsen:EnumerationContext—The latest version of this object that the client has received from the server.
- idPrefix—String that is added to the beginning of the internally generated numeric to form the unique NBI identifier. A null or empty string is valid, resulting in an NBI identifier that is just the internally generated numeric.

Returns:

Standard WS-Enumeration Renew response message.

Throws:

Remote Exception—If the server cannot execute the NBI request.

SyncReturn:

Returns the new expiration date.

getStatusOp Java Signature

Updates the ApiEnumerator object.

Syntax:

public ResponseReturn getStatusOp (CUPMServiceStub stub, wsen:EnumerationContext enumerationContext);

Parameters:

wsen:EnumerationContext—The latest version of this object that the client has received from the server.

Returns:

Standard WS-Enumeration getStatus response message.

Throws:

RemoteException—If the server cannot execute the NBI request.

SyncReturn:

Returns the current expiration date.

releaseOp Java Signature

Tells the server that the client is done with the data and it may be purged.

Syntax:

public ResponseReturn release (CUPMServiceStub stub, wsen:EnumerationContext enumerationContext);

Parameters:

wsen:EnumerationContext—The latest version of this object that the client has received from the server.

Returns:

WS-Enumeration standard Release response message.

Throws:

RemoteException—If the server cannot execute the NBI request.

SyncReturn:

A response with no parameters.

Prime Collaboration Provisioning NBI Implementation of WS-Notification

A Prime Collaboration Provisioning client submits an NBI request to the Prime Collaboration Provisioning server. The two platforms are Web Services peers. The client is a NotificationConsumer and is subscribed to the Web Services resource that is the Prime Collaboration Provisioning server. The Prime Collaboration Provisioning server is a NotificationProducer. It has only one topic and is viewed as only one Web Services resource. When the Prime Collaboration Provisioning server completes the NBI request, it will publish a notification with the NBI request results. As each client subscribes to the server, a filter is implemented such that when the Prime Collaboration Provisioning server publishes the NBI request result notification, only the client that submitted that NBI request will receive the response. There is a one-to-one relationship between NBI requests accepted by the server and NBI request result notifications sent to the client.

There are no cases where the Prime Collaboration Provisioning server will send a message to the client unsolicited.

Notification Consumer Service

The Prime Collaboration Provisioning NBI SDK provides a notification consumer service to receive the request notifications. The notification consumer service is written in Java and designed to be deployed in an axis2.war file, running under Tomcat.

The default functionality for this service is to receive the notification on port 8080 and display the notification in the Tomcat console window.

The notification software includes a method to write the notification to disk. The directory to write the notifications in is set by the environment variable consumer.home. The default value used if this environment variable is not set is \NotificationConsumer. If you want to write code to process the notification differently, you can comment the call to this method out of the Java code. The call is in the NotificationConsumerServiceSkeleton.java file.

The ability to write the notification to disk is required, if you want to use the auto pull feature in the Perl clients for list subscriber or list Domain.

Appendix A: Sample Clients

This section describes how to use the source code, jar files, and XML files to build sample clients. For detailed information, see the following sections:

- Introduction
- Setting Up the Prime Collaboration Provisioning NBI SDK Development Environment
- Setting Up the Sample Java Clients
- Setting Up the Perl NBI Client

Introduction

Using Ant, you will compile and generate two Prime Collaboration Provisioning NBI clients. The clients use environment variables to specify the server URL to send the request to.

A simple notification consumer service is provided. Using Ant, you will compile and generate archives that can be installed in an AXIS2.war file that has been installed in Tomcat.

You will also be able to package the clients, notification consumer, and tools into a zip file that can be installed on Windows XP platforms.

Setting Up the Prime Collaboration Provisioning NBI SDK Development Environment

In your Prime Collaboration Provisioning installation, you will find a sample client-development directory at CUPM\sep\ipt\nbi-sdk. You may either work in the directory or copy it to another Windows XP platform. This platform must have Java 1.6 and Ant 1.6.5 installed. If you intend to run the Notification Consumer, you will also need to install Tomcat and AXIS2.

Overview of Prime Collaboration Provisioning NBI SDK Directory

This section lists the directories included in the Prime Collaboration Provisioning NBI SDK. After you download and unzip the SDK, you will see the following directories under the ...\nbi-sdk directory:

- doc—User documentation.
- html-doc—For each NBI XSD file, this directory contains html documentation generated from the annotation fields in the XSD file. In addition, there are graphic descriptions of all NBI complex objects.

The documentation describes all complex objects used by the Prime Collaboration Provisioning NBI. For top level objects, it discusses which attributes are the keys and which attributes must be set in the create request for that object. There are also graphic representations of the objects.

- productcatalog—Contains the XML file version of the complete Prime Collaboration Provisioning product catalog. In the product catalog directory, there are two subdirectories: the schema subdirectory that has an XSD file that defines all products supported by Prime Collaboration Provisioning, and the metadata subdirectory that contains XML definitions for every product that can be ordered using Prime Collaboration Provisioning.
- sample—Contains the Java, bat, and XML files for building and running sample clients. Also included are sample create requests for configuration objects.
- wsdl-xsd—Contains all WSDL and XSD files that define the Prime Collaboration Provisioning NBI.

Setting Up the Sample Java Clients

After setting up your environment, either on your Prime Collaboration Provisioning system or on another system, go to the CUPM\sep\ipt\nbi-sdk\sample directory.

The build.xml file in this directory creates the subdirectories output and image. The output directory contains all generated class, jar, zip, and other targets. The image directory is the installable image that can be run from here when referenced with the environment variable CUPM_NBI_SDK_COMMAND_HOME. The image directory is also what is condensed into the zip file for installation on other platforms.

Use the build-install-all target to run all the Ant targets required to prepare the clients and to prepare and deploy the notification consumer service.

The build.xml file provides the following targets:

- build-install-all—Compiles and builds clients, and compiles, builds, and installs notification consumer.
- build-clients—Only builds clients.
- compile-clients—Java compilation.
- lib-clients—Creates jar files for clients.
- image-clients—Adds the clients to the installable image directory.
- build-consumer—Builds only the notification consumer service.
- compile-consumer-schema
- compile-consumer-src
- lib-consumer

- image-consumer

After you have built an image, or installed an image on another platform, run the following Ant commands from the CUPM\sep\ipt\nbi-sdk\sample directory:

- setup-axis2-tomcat—Generates an AXIS2.war file from a new AXIS2 installation, then copies and expands that AXIS2.war file into a Tomcat installation.
- deploy-consumer—Installs the notification consumer in a version of Tomcat with AXIS2 installed and expanded.

Running SDK Sample Clients

After you have prepared your image directory, run the SDKSetup.bat command to set the appropriate environment variables and test the connection with the simple ping command. If you prefer to manually prepare your environment, set the environment variable.

You now have access to the following SDK commands:

- **SdkQueryEnvVar**—Displays all Environment Variables that are used by the SDK clients, along with their current settings.
- **ValidateRequest <XML-File>**—Validates an XML file that contains any NBI request. The SAX parser performs the validation against the appropriate XSD file.
- **PingRequest <Optional Arguments>**—Sends a ping request to the specified Prime Collaboration Provisioning server and sets the EPR to the notification consumer service.

Note**:** The Server URL and Notification Consumer URL can be specified by the environment variables, or by arguments on the command line. An argument of -h provides a description of the command arguments.

- **SendXmlRequest <XML-File>**—This client reads an XML file. It optionally substitutes the EPR in the file with the one specified by the environment variables. It optionally substitutes the setting for the idPrefix. It then sends the request and prints the response. If the request sends a notification, it is printed in the Tomcat command window by the notification consumer service.

Environment Variables

The sample NBI clients allow you to set the server information and notification either on the command line, or by using environment variables. The environment variables enable you to issue a set of requests to one server and receive notifications at one URL. It may be easier for you to set parameters using environment variables instead of setting them every time on the command line.

Table 10: Environment Variables

Environment Variable	Default	Purpose/Notes
server.ipaddress	localhost	IP address of the Prime Collaboration Provisioning server that you wish to send your requests to.
server.port	80	Port configured for the graphical user interface and Web Services access for Prime Collaboration Provisioning. **Note** : If you receive transport errors, your server may not be using port 80. If a server is installed on a platform where port 80 is in use, it will select another port, often port 1024.
server.namespace	axis2/services/CUPMServices	The configured service on Prime Collaboration Provisioning. It should never be changed.
server.url	—	If set, it is used for the full URL name. If it is not set, the server.ipaddress, port, and namespace are combined to create the server URL.
notification.ipaddress	localhost	The IP address of the notification consumer, where the NBI results are sent.
notification.port	8080	The port of the notification consumer URL.
notification.namespace	axis2/services/NotificationConsumerService	The namespace of the server. This is the name provided in the samples. If you write your own, you would put your name here.
notification.url	—	If set, it is used for the full URL name. If it is not set, the notification.ipaddress, port, and namespace are combined to create the server URL.
cupm.user	—	The Prime Collaboration Provisioning account with administrative privileges that is used as the credentials for Prime Collaboration Provisioning NBI access. **Note**: The SdkSetup.bat file assumes this name is pmadmin. If you set this account to a different name during installation, you will need to manually set it to the correct ID.
cupm.password	—	The password for the Prime Collaboration Provisioning account with administrative privileges that is used as the credentials for Prime Collaboration Provisioning NBI access. **Note**: The SdkSetup will expect this password as an argument, so that it can set it for you.
server.protocol	http	Determines if the request/response transaction is in http or https. Setting this environment variable to https will implement https transmission, but will not check the certificate unless the https.certificate environment variable is set. **Note** : Prime Collaboration Provisioning server must be configured for https. For instruction, see Installation Guide for Prime Collaboration Provisioning.
https.certificate	False	Boolean setting, true or false. A null setting is considered false. If set to true, the certificate from the server is validated as part of each request transaction. Installing the certificate from the server is a prerequisite for using this feature. It is recommended that you download and run the Sun Microsystems command InstallCert.java.
request.idprefix	—	There is an optional NBI parameter for all asynchronous requests to add to the beginning of the NBI ID. It is specified in the XML request file. If you set this environment variable, it will override the value in your XML file.

XML Sample Files

Included are examples of XML requests to create, update, get, delete, and list some network objects. Simple Orders to configure phones and lines are included.

Note: These requests will require editing to match your configuration.

The sample XML files are commented such that you can update them to match your environment. For example, for create device, at the minimum, you would need to set the IP address and the credentials to match your device.

Installing and Running the Prime Collaboration Provisioning NBI SDK

This section lists the minimal steps to build and test the Prime Collaboration Provisioning NBI clients.

1. Install Java JRE or JDK 1.5 on your system.
    a. Install Java from the zip file, or install it using a different method.
    b. Set the JAVA_HOME parameter to reference the Java 1.6 directory.
    c. Add %JAVA_HOME%\bin to your path.

Note: If you intend to install Prime Collaboration Provisioning on the same platform as the SDK, you can skip this step and use the Java version provided with Prime Collaboration Provisioning.

2. Install Ant 1.6.5 on your system:
    a. Download the zip file.
    b. Unzip the file.
    c. Set the ANT_HOME environment variable to the unzipped directory.
    d. Add %ANT_HOME%\bin to your path.
3. Install Axis2 1.3:
    a. Download the zip file.
    b. Unzip the file.
    c. Set AXIS2_HOME environment variable to the unzipped directory.
    d. Add %AXIS2_HOME%\bin to your path.
4. Install Tomcat 5.5:
    a. Download the zip file.
    b. Unzip the file.
    c. Set CATALINA_HOME environment variable to the unzipped directory.
    d. Add %CATALINA_HOME%\bin to your path.
5. Install Prime Collaboration Provisioning. You may install Prime Collaboration Provisioning either on the SDK system or on a different system. For simple testing, development, and understanding of the Prime Collaboration Provisioning NBI, the same system may be appropriate. For any significant development, it is recommended that you install Prime Collaboration Provisioning on a different system.
    - Run the Prime Collaboration Provisioning setup file (starts the installShield wizard). This will install Java 1.6 and set the JAVA_HOME environment variable.

Note: The Prime Collaboration Provisioning server will be running after installation.

6. Install the Prime Collaboration Provisioning NBI SDK:
    a. Unzip the cupm-nbi-sdk.zip file in any directory.
    b. CD to the sample subdirectory (...\nbi-sdk\sample).

Note: If you wish to build a client with Java 1.5 or an earlier version, you must rebuild the cupm-nb-api.jar file with your chosen Java compiler. The ant target rebuild-jar will compile the Java code in the src directory and recreate a new version of this file in the lib directory. Run the command ant rebuild-jar before proceeding.

    c. Run ant build-install-all. This performs the following:
        - Compiles and builds two sample clients.
        - Compiles and builds an XML request validator utility.
        - Compiles and builds a notification consumer service AAR file.
        - Creates the axis2.war file.
        - Installs and expands axis2.war in tomcat.
        - Deploys the notification consumer service in axis2 in Tomcat.
    d. Run %CATALINA_HOME%\bin\startup. It starts Tomcat.
    e. Run SdkSetup.bat <pmadmin-password>. Run this bat file with your pmadmin user ID password as the first argument. This bat file performs the following:
        - Sets the environment variable CUPM_NBI_SDK_COMMAND_HOME to the SDK command image built in the previous step (...\nbi-sdk\sample\image\CupmNbiSdkCommands).
        - Adds CUPM_NBI_SDK_COMMAND_HOME to the path.
        - Sets the following SDK environment variables:
            - server.ipaddress=localhost
            - notification.ipaddress=localhost
            - cupm.user=pmadmin
            - cupm.password=<bat-parameter>
        - Runs the PingRequest client. As this runs, the data from the ping response is displayed. It shows the Prime Collaboration Provisioning version and the Prime Collaboration Provisioning NBI version. Also, the notification server status string is displayed in your tomcat window.

Note: The SdkSetup.bat is designed for the SDK and Prime Collaboration Provisioning to be installed on the same system and to use the default values for Prime Collaboration Provisioning's account name and ports.

If any of the following apply to your installation of Prime Collaboration Provisioning, you will need to set the environment variables to your customized settings:

- Prime Collaboration Provisioning is installed on a system other than the SDK.
- The Prime Collaboration Provisioning account is not pmadmin.
- Port 80 is not used for the server.
- Port 8080 is not used for notification consumer.
- You are using your own notification consumer service.

If you have a configuration other than basic, run SdkSetup.bat and let it fail. Then use the command SET to set the environment variables to match your configuration.

a. Run SendXmlRequest xml\CreateDevice.xml. Sends the XML file CreateDevice.xml in the XML subdirectory to the server. This creates a fictitious device with IPT capabilities. As this runs, a response XML file appears and after a few seconds, the notification result XML file appears in the Tomcat window.

   To create a real device, you must first edit the CreateDevice.XML file. Change the IP address and credential tags to match the real device.

b. Run SendXmlRequest xml\CreateDomain.xml. Sends the XML file CreateDomain.xml in the XML subdirectory to the server. This creates a Domain. As this runs, a response XML file appears. After about a minute, the Notification Result XML file appears in the Tomcat window.

c. Run SendXmlRequest xml\CreateServiceArea.xml. Sends the XML file CreateServiceArea.xml in the XML subdirectory to the server. This creates a Service Area within the Domain. As this runs, a response XML file appears. After a few seconds, the Notification Result XML file appears in the tomcat window.

d. Run SendXmlRequest xml\CreateSubscriber.xml. Sends the XML file CreateSubscriber.xml in the XML subdirectory to the server. This creates a subscriber within the Domain. As this runs, a response XML file appears. After a few seconds, the Notification Result XML file appears in the Tomcat window.

e. Run SendXmlRequest xml\CreateSubscriber2.xml. Sends the XML file CreateSubscriber2.xml in the XML subdirectory to the server. This creates a second subscriber within the Domain. As this runs, a response XML file appears. After a few seconds, the Notification Result XML file appears in the Tomcat window.

SDK Commands

Once you have installed the SDK and run SdkSetup, you will have access to the SDK commands from that command window. They can be run from any directory.

The following are the SDK commands:

- SendXmlRequest—Reads an XML file from disk. The first argument is the filename of the XML file, and the full path filename if this file is not in the current directory. This command will use the environment variable settings to modify the XML file before sending. You can also set or override any current settings from the command line. Enter -h as the first argument for a display of the usage of this command.
- ValidateRequest—Reads an XML file from disk. Without making any substitutions, the XML file is validated with the SAX parser against the appropriate XSD file to confirm that this is a valid XML file. File is valid appears if the file passes validation. If validation fails, a SAX parser exception is thrown, describing the first issue encountered. Enter -h as the first argument for a display of the usage of this command.
- PingRequest—Sends the ping request only. Nothing is read from disk. The instance is created and populated within Java. The ping request is used to test connectivity and credentials from the client to the server, and to test connectivity from the server to the notification consumer. Enter -h as the first argument for a display of the usage of this command.
- SdkQueryEnvVar—Displays all environment variables that are used by the other commands, and their current settings.
- SdkVersion—Displays the date and time that the SDK commands were built.

NBI Prepopulation Requirements

The ListProductAttributeChoice request is often referred to as the prepopulation NBI. It is designed to return choice lists for attributes in a product. The choice lists can be either complete choice lists or context-based choice lists dependent on other settings. The choice lists returned through the NBI require some additional information to be provided in the request.

Table 11 lists any additional requirements, limitations, and restrictions for returning choice lists for the attributes in a product.

Table 11: Choice List of Attribute Requirements

Product	Attribute	NBI Specifics
• Phone • Line • LineOnSharedPhone • EnableCUPCLicense	Type	All types are returned. No filtering based on roles or Service Area. This information is different from what is displayed in the Prime Collaboration Provisioning user interface.
Phone	• emenable	Type must be specified.

	• usedummymacaddress	
Phone	pbt	Type and protocol must be specified.
Line	Lineposition	Requires SelectedPhone attribute to be passed in for Line. For Line On Shared Phone, it requires target phone to be passed in.
Line	SelectedPhone	Not supported
• Line • EM Line • Line On Shared Phone	Directorynumber	Not supported
Line On Shared Phone	targetphone	Not supported
EM Access	pbt	Requires protocol as additional information.
EM_Line	SelectedEM_Access	Not supported
Voicemail	SelectedLine	Not supported
Email	SelectedVoicemail	Not supported
UnifiedMessaging	SelectedEmail	Not supported

Setting Up the Perl NBI Client

A sample Perl client is included in the subdirectory perl/pm.

Requirements

Perl 5.10.0 is required. The NBI client was developed and tested with ActivePerl, which offers free downloads of the ActivePerl interpreter. The ActivePerl-5.10.0.1005-MSWin32-x86-290470.msi file was used. You should run the installation file, rather than just downloading the zipped version.

Common Properties

The Perl client uses the same environment variables that the Java client does for the Prime Collaboration Provisioning Server EPR and credentials, and for the notification EPR. Environment variables can be used throughout the session for every request, or can be specified or overwritten for one request from the command line.

XML Template Usage

The Perl client uses XML templates for sending the requests. Each template contains the full SOAP envelope for the command. Keywords are substituted by the user-specified values, resulting in a valid XML request that is then sent to the server.

Table 12 - Keywords Supported by the NBI Perl Client

Property Name or Source of Data	Perl Keyword	Notes
server.url	SERVER_URL	If you set server.url, then it is substituted as specified. If server.url is not specified, then the URL is constructed from the other server properties.
server.protocol	SERVER_PROTOCOL	Only http is supported for the PERL Client example. It could be extended to support https.
server.ipaddress	SERVER_URL	—
server.port	SERVER_PORT	—

server.namespace	SERVER_NAMESAPCE	Axis2/services/CUPMService is the only valid setting.
notification.url	NOTIFICATION_URL	Same logic as server.url.
notification.ipaddress	NOTIFICATION_IPADDRESS	—
notification.port	NOTIFICATION_PORT	—
notification.namespace	NOTIFICATION_NAMESPACE	—
cupm.user	CUPM_USER or USERNAME	—
cupm.password	CUPM_PASSWORD or PASSWORD	—
cupm.password is initial source of data	ENCODED_PASSWORD	The cupm.password after encoding with base 64 encryption.
form command line	REQUEST_NAME	Name of the NBI request.
object.name	OBJECT_NAME	Designed to be request sensitive. For a pull request, the object name defines the NBI ID of the list to retrieve instances.
pull.count (defaults to 1 if not specified)	PULL_COUNT	—

Supported Requests

The sample Perl client supports the following requests:

- ping
- pull
- listDomain
- listSubscriber

The architecture allows extension of additional commands by adding a Perl Module (.pm) file to the subdirectory modules, and a corresponding XML template file to the subdirectory XML/CUPMRequest.

Usage (Sample Session)

The Perl client is run from the command line. It must be run from the perl/pm subdirectory, unless that subdirectory is added to the system path environment variable.

The command is pm.pl. Enter the command with no arguments or with –h for online help.

The command takes one argument, the request name. Additional properties can be assigned with the –D prefix.

Sample Session

The following commands will ping a server and then retrieve a Domain list from it:

1. Set up the environment variables:
   - set server.ipaddress=1.2.3.4
   - set cupm.user=pmadmin
   - set cupm.password=yourpassword
   - set notification.ipaddress=5.6.7.8
2. Run the perl client: pm.pl ping. Sends the ping request. The Prime Collaboration Provisioning version and the Prime Collaboration Provisioning NBI version are returned in the response and displayed.
3. Run pm.pl ping -Dserver.ipaddress=11.22.33.44 -Dcupm.password=diffpassword. Sends a ping request to a different server, temporarily overriding the environment variable settings.
4. Run pm.pl listDomain. Sends the listDomain command. The NBI ID is returned in the response and displayed.
5. Run pm.pl pull -Dobject.name=<NBI-ID-RETURNED-BY-ABOVE-LIST>. Returns the first element of the list of Domains.
6. Run pm.pl pull -Dobject.name=<NBI-ID-RETURNED-BY-ABOVE-LIST> -Dpull.count=1000. Returns the remaining objects on the list, up to 1000.
7. The listDomain and the listSubscriber requests provide the functionality to automatically pull all the items in the list immediately after the list is generated. This is triggered by setting the pull.count environment variable to a positive value.

   For example: pm.pl listSubscriber -Dpull.count=5

   This sends a request for a list of all the subscribers in the system. It waits for the notification consumer to write the notification to disk. Then it pulls the list back in increments of five until the end of sequence attribute is received.

   Note: For this functionality to work, you must compile the NotificationConsumerService after uncommenting the call to the method saveMessageResult.

Appendix B: Troubleshooting

This appendix provides troubleshooting information.

FAQs

When I submit a large number of NBI requests rapidly to the server, after a while, I start seeing throttling exceptions and my requests are rejected by the server. Why is this?

The Prime Collaboration Provisioning Server only allows 20 requests to execute at one time. There is no queuing on the server; requests exceeding the limit are rejected with a throttling exception.

When I use the NBI syncDomain request, I receive a notification with a fault that no Domain synchronization rules are configured. What do I have to do?

Before you can synchronize a Domain, you must set the Domain synchronization rules. There is no NBI request for setting the Domain synchronization rules; you must do so through the Prime Collaboration Provisioning user interface.

To set up the Domain synchronization rules:

1. In the Prime Collaboration Provisioning user interface, select Advanced Setup > Policies > Rules.
2. Select a Domain.
3. Click Configure Domain Sync Rules.
4. Select the synchronization rules that you want to be applied to the Domain.

On the Rules page, if you select Customer Domain Template and Configure the Domain Sync Rules, when a Domain is created, the rules will be copied to the Domain.

When I try to delete a device, Domain, or Service Area, I get an exception thrown that I need to be in maintenance mode. What do I have to do?

Before you can delete a device, Domain, or Service Area, you must change Prime Collaboration Provisioning to maintenance mode. There is no NBI request for changing the operational mode; you must do so through the Prime Collaboration Provisioning user interface.

To put Prime Collaboration Provisioning into maintenance mode:

1. In the Prime Collaboration Provisioning user interface, select System Administration > Maintenance Mode.
2. Select Enter Maintenance Mode.

When I send a submitOrder request to the server, I get the exception that I am in maintenance mode. What do I have to do?

Infrastructure configuration orders and subscriber orders can be submitted only in normal mode (not maintenance mode).

To put Prime Collaboration Provisioning into normal mode:

1. In the Prime Collaboration Provisioning user interface, select System Administration > Maintenance Mode.
2. Select Exit Maintenance Mode.

I cannot use Prime Collaboration Provisioning to disable certain infrastructure products on Cisco Unified Communications Manager. What is preventing me?

Some values in Cisco Unified Communications Manager, once selected and set to true in Prime Collaboration Provisioning, cannot be changed to false. This applies to values that are displayed as check boxes in Cisco Unified Communications Manager.

Following are some examples:

- In Voice Mail Profile, the attribute IC_isDefault corresponds to the check box Make this the default Voice Mail Profile for the System. The attribute IC_isDefault cannot be changed from true to false.
- In Voice Mail Pilot, the attribute IC_isDefault cannot be changed from true to false.
- In H323 Gateway, the attribute IC_mtpRequired cannot be changed from true to false.

I want to remove one subscriber role from a Service Area without deleting and creating an object. How do I do that?

The metadata action property allows you to remove one setting from a property that takes multiple values.

For the NBI request updateServiceArea request, specify the following for the metadata value:

- The attribute name you want to update.
- The attribute value you want to remove.
- The property action setting of REMOVE.

I have created multiple phones and multiple lines in one order. When I send the request with submitOrder, how can I associate the lines to the appropriate phones?

Orders have sequence numbers for each detail. When adding a service based on a service previously added in the order, set the assocSequence value to the sequence number to make the association.

After synchronizing a device, I am not seeing all the directory numbers that are available on the Cisco Unified Communications Manager in Prime Collaboration Provisioning. Why is that?

If a directory number on Cisco Unified Communications Manager has not been assigned to a subscriber in Cisco Unified Communications Manager, it will not be uploaded to Prime Collaboration Provisioning during the syncDevice request.

I sent a list request with a filter for multiple attribute values using the logical OR. My resulting list only uses the first attribute value I specified.

The attribute value type in the filter is an array. However, the array is designed to store multiple values for one attribute setting and not designed to store all the attribute values that you wish to apply to the logical OR in the query. Each value for the OR must be specified as a unique attribute name/value pair.

For example, consider the filter to return all devices that have a capability of either Cisco Unity or Cisco Unity Connection.

The following is incorrect:

```
<filter>
<cmn:objectSelectionCriteria>
<cmn:selectionCriteria>
<cmn:item>
<cmn:attributeName>capabilityName</cmn:attributeName>
<cmn:attributeValue>
<co-v1-3:item>CiscoUnity</co-v1-3:item>

<co-v1-3:item>CiscoUnityConnection</co-v1-3:item>

</cmn:attributeValue>
</cmn:item>
</cmn:selectionCriteria>
<cmn:flag>OR</cmn:flag>
</cmn:objectSelectionCriteria>
<cmn:returnDataCriteria>
<cmn:returnDataType>ALL</cmn:returnDataType>
</cmn:returnDataCriteria>
<cmn:associatedObjectSelectionCriteria/>
<filter>
```

The following is correct:

```
<filter>

<cmn:objectSelectionCriteria>
<cmn:selectionCriteria>
<cmn:item>
<cmn:attributeName>capabilityName</cmn:attributeName>
<cmn:attributeValue>
<co-v1-3:item>CiscoUnity</co-v1-3:item>
</cmn:attributeValue>
</cmn:item>
<cmn:item>
<cmn:attributeName>capabilityName</cmn:attributeName>
<cmn:attributeValue>
<co-v1-3:item>CiscoUnityConnection</co-v1-3:item>
</cmn:attributeValue>
</cmn:item>
</cmn:selectionCriteria>
<cmn:flag>OR</cmn:flag>
</cmn:objectSelectionCriteria>
```

```
<cmn:returnDataCriteria>
<cmn:returnDataType>ALL</cmn:returnDataType>
</cmn:returnDataCriteria>
<cmn:associatedObjectSelectionCriteria/>
</filter>
```

When I send a get request or a getOrderSummary request, or any request that returns data back in a notification that is contained in a cupmApiList, how do I extract the individual objects from the list?

There are several ways to do this in Axis2-supported Java. See the following Java code snippets for examples.

```
// get the arrayOfObject object out of the list
org.ossj.xml.common.v1_3.ArrayOfObject arrayOfObject =
cupmApiList.getDataObject();


// process each object in the list int sizeArray =
arrayOfObject.sizeOfItemArray();
for ( int ix = 0 ; ix < sizeArray ; ix++ ) { XmlObject xmlObject =
arrayOfObject.getItemArray(ix); processObject(xmlObject); }


void processObject(XmlObject xmlObject) {
// Ror each object in the list, the class, a typed object, and/or the
// node name can be used to determine and cast the object on the list.
// Some objects can be done multiple ways, others, sent back as
// documents, must use the typed object OMElement element = new


StAXOMBuilder(xmlObject.newXMLStreamReader()).getDocumentElement();
XmlOptions options = new XmlOptions();
options.setLoadReplaceDocumentElement(element.getQName());
typedXmlObject = XmlObject.Factory.parse(xmlObject.newInputStream(), options);
String nodeNameKey = getTypeFromListItem(xmlObject);
// nonimpl way to identify node Class xmlClass = xmlObject.getClass();
if (
xmlClass.equals(com.cisco.cucms.cupm.nbi.v1.impl.CupmApiResultValueImpl.class) )
{
// if list contains a apiResult, you may use the result in its value state,
// or you can create a document class and work with the result for more
functionality
CupmApiResultValue cupmApiResult = (CupmApiResultValue)
xmlObject; CupmApiResultDocument cupmApiResultDocument =
CupmApiResultDocument.Factory.newInstance();
cupmApiResultDocument.setCupmApiResult(cupmApiResult); } else if (
xmlClass.equals(com.cisco.cucms.cupm.nbi.v1.impl.DeviceValueImpl.class) ) {


// if list contains a device, you may use the device in its value state,
// or you can create a document class and work with the device for more
functionality
```

```
DeviceValue device = (DeviceValue)xmlObject; DeviceDocument deviceDocument =
DeviceDocument.Factory.newInstance(); deviceDocument.setDevice(device);
// This also applies to domain, serviceArea, Subscriber, Order, InventoryItem,
Product,
// ServiceDetail, PhoneInfo, String, CupmApiFaultType, CupmApiList } else if (
nodeNameKey.equalsIgnoreCase("EnumerationContextType") ) {
// For Enumeration Object, better to use node name instead of support class so
client
// is not limited to the Axis2 implementation of Enumeration
EnumerationContextType
enumerationContext = (EnumerationContextType)xmlObject;
EnumerationContextDocument
enumerationContextDocument = EnumerationContextDocument.Factory.newInstance();
enumerationContextDocument.setEnumerationContext(enumerationContext); } else if (
typedXmlObject instanceof OrderSummaryDocument ) {
// OrderSummary comes back as a document so must used the typedXmlObject for
casting
OrderSummaryDocument orderSummaryDocument = (OrderSummaryDocument)typedXmlObject;
if ( ps
!= null ) ps.println(orderSummaryDocument.toString()); } else if ( typedXmlObject
instanceof com.cisco.cucms.cupm.nbi.v1.KeywordListDocument ) {
// KeywordList comes back as a document so must used the typedXmlObject for
casting
com.cisco.cucms.cupm.nbi.v1.KeywordListDocument keywordListDocument =
(com.cisco.cucms.cupm.nbi.v1.KeywordListDocument)typedXmlObject; if ( ps != null
)
ps.println(keywordListDocument.toString()); } else { System.out.println("WARNING
- unknown
class = " + xmlClassName + " node name = " + nodeNameKey ); } }
```

When I run getOrder or listOrder with a return attribute of all, some of the settings do not match the attribute values that were passed in the initial submitOrder request.

When the order is retrieved from the Prime Collaboration Provisioning database, some settings (in particular, boolean settings) have their internal attribute values returned.

When I create a createServiceArea request or a submitOrder request, an attribute that looks correct to me is rejected. Why is that?

Attribute names are case sensitive. Make sure that your name exactly matches the XSD string. Also, double check the to make sure that the attribute definition is in the right object. For example, device and device capability are different levels in the createDevice request.

When I create a createServiceArea request with VoiceGatewayReference, the request is rejected with message stating, “Invalid VoiceGatewayReferenceName,” but the specified VoiceGatewayReference exists in CUPM. Why is that?

VoiceGatewayReferenceName will be considered as valid only if the Slot and Subunit properties are configured for the VoiceGatewayReference. Ensure that the VoiceGatewayReference name you are specifying in the request has a slot and subunit configured.