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, page A-1
- Setting Up the Provisioning Manager NBI SDK Development Environment, page A-1
- Setting Up the Sample Java Clients, page A-2
- Setting Up the Perl NBI Client, page A-9

Introduction

Using Ant, you will compile and generate two Provisioning Manager 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 Provisioning Manager NBI SDK Development Environment

In your Provisioning Manager 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 Provisioning Manager NBI SDK Directory

This section lists the directories included in the Provisioning Manager 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 Provisioning Manager 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 Provisioning Manager 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 Provisioning Manager, and the metadata subdirectory that contains XML definitions for every product that can be ordered using Provisioning Manager.
- 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 Provisioning Manager NBI.

Setting Up the Sample Java Clients

After setting up your environment, either on your Provisioning Manager 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 Provisioning Manager 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 A-1 lists the supported environment variables.

Table A-1 Environment Variables

Environment Variable	Default	Purpose/Notes
server.ipaddress	localhost	IP address of the Provisioning Manager server that you wish to send your requests to.
server.port	80	Port configured for the graphical user interface and Web Services access for Provisioning Manager. **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 Provisioning Manager. 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 Provisioning Manager account with administrative privileges that is used as the credentials for Provisioning Manager 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 Provisioning Manager account with administrative privileges that is used as the credentials for Provisioning Manager NBI access. **Note** The SdkSetup will expect this password as an argument, so that it can set it for you.

Table A-1 Environment Variables (continued)

Environment Variable	Default	Purpose/Notes
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** Provisioning Manager server must be configured for https. For instruction, see *Installation Guide for Provisioning Manager*.
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 Provisioning Manager NBI SDK

This section lists the minimal steps to build and test the Provisioning Manager 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 Provisioning Manager on the same platform as the SDK, you can skip this step and use the Java version provided with Provisioning Manager.

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 Provisioning Manager 2.0. You may install Provisioning Manager 2.0 either on the SDK system or on a different system. For simple testing, development, and understanding of the Provisioning Manager NBI, the same system may be appropriate. For any significant development, it is recommended that you install Provisioning Manager on a different system.
   - Run the Provisioning Manager 2.0 setup file (starts the installShield wizard). This will install Java 1.6 and set the JAVA_HOME environment variable.

Note The Provisioning Manager server will be running after installation.

6. Install the Provisioning Manager 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 Provisioning Manager version and the Provisioning Manager NBI version. Also, the notification server status string is displayed in your tomcat window.

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

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

- Provisioning Manager is installed on a system other than the SDK.
- The Provisioning Manager 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.

f. 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.

g. 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.

h. 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.

i. 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.

j. 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 A-2 lists any additional requirements, limitations, and restrictions for returning choice lists for the attributes in a product.

Table A-2 Choice List for 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 Provisioning Manager user interface.
Phone	• emenable • usedummymacaddress	Type must be specified.
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 Provisioning Manager 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 A-3 lists keywords supported by the NBI Perl client.

Table A-3 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 Provisioning Manager version and the Provisioning Manager 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.