Dialogic®

Tech Note

Using the Dialogic® PowerMedia™ Extended Media Server (XMS) 2.0 RESTFul Management API - A Java Demo

Using the Dialogic® PowerMedia™ Extended Media Server (XMS) 2.0 RESTFul Management API - A Java Demo

Tech Note

Introduction

Dialogic® PowerMedia™ Extended Media Server (PowerMedia XMS) is usually configured and managed through an interactive web-based GUI. However, there may be circumstances which dictate that system management be done in a more automated or distributed manner. To address such circumstances, the PowerMedia XMS RESTful Management API provides an alternate way of performing system management tasks.

The RESTful Management API uses conventional RESTful methods of POST – add a new resource, PUT – modify an existing resource, GET – return information about a resource, and DELETE – remove a resource. It uses secure HTTP, or HTTPS, as its transport so that information about PowerMedia XMS server configuration cannot be intercepted and read.

While the RESTful messages themselves are relatively straightforward, the infrastructure surrounding their use can be complicated. This is especially true in the use of HTTPS and security certificates. In particular, obtaining, installing and using test and production certificates can be confusing, and HTTPS itself requires some additions that are not needed with unsecure HTTP. In addition, JavaScript Object Notation (JSON) is used in the RESTful message payloads. It is possible to parse and build these payloads manually, but there are libraries available that can be used to help do this more easily.

The main purpose of this demo is to provide examples of the four (4) HTTP methods used by the RESTful Management API. Other key points are the use of the Java secure socket library classes to do HTTPS requests and the use of a JSON parser to encode JSON request payloads and parse JSON response payloads. In addition, the use of binary payloads for certain commands is illustrated.

While any language and programming environment may use the RESTful Management API, Java was chosen for this demo. It is a well-known, platform independent, object-oriented language, and the techniques used here will have equivalents in other languages such as C++ or Python. The demo does not use a Graphical User Interface (GUI), as its intent is not to show how to build a replacement for the existing PowerMedia XMS GUI. Rather, it runs through a series of management commands that illustrate most of the types of functions available in the API. The output for the commands is logged, and can be examined after the management sample is run. The full reference that describes all the RESTful management resources may be found here – https://www.dialogic.com/webhelp/XMS/2.0/XMS_RESTfulMgmtAPIDev/default.htm

HTTPS and Security

Generally, web browsers use their own certificate store, or a system-wide store, and have ways to handle security certificate exceptions. With an HTTPS Java application, security must be handled as part of the app itself.

Java's certificate store is located in $JAVA_HOME/jre/lib/security, where $JAVA_HOME points to the Java installation in use. Changes must be made to the default certificate store in a manner consistent with the type of security certificate used. PowerMedia XMS comes with a "self-signed" certificate which may be used for testing purposes. A legitimate security certificate should be obtained from one of the many certificate signing authorities for production use.

In either case, the Java utility "keytool" (located in $JAVA_HOME/bin) is used to manipulate the certificate keystore.

Using the Self-Signed Certificate

The self-signed certificate "xms.crt" accompanies the PowerMedia XMS installation. This certificate is intended only for test purposes and should **not** be used in a production situation where true HTTPS security is needed. Besides having no certificate authority to verify it, it also requires an addition to the HTTPS request to disable hostname verification – the process by which the client verifies that it is receiving information from the proper server.

The following steps are needed to install the self-signed certificate in a Java keystore:

1. Copy the file /etc/lighttpd/ssl/xms.crt from the PowerMedia XMS installation to the directory containing the Java keystore on the system on which the Java demo will be run. For Java 1.7, update 9, this was
   C:\Program Files\Java\jdk1.7.0_09\jre\lib\security
2. Create a new keystore with a single certificate in it:

```
keytool -import  -alias xms_hostname.xms_domainname -file xms.crt
        -keystore xmscert
```

   The default password for the keystore is "changeit". Answer yes to "Trust this certificate?"
3. When running the demo, supply the keystore name as –keystore "C:/Program Files/ Java/jdk1.7.0_09/jre/lib/security/xmscert" Note that forward slashes are used, even for a Windows system. If no keystore name is given, it will use the default location just given.

Using a Legitimate Certificate

All web technologies are able to make use of secure HTTPS connections, and do so in a similar manner. A good tutorial on using HTTPS with Java, is here: http://docs.oracle.com/javaee/1.4/tutorial/doc/Security6.html#wp80737

For a secure PowerMedia XMS RESTful management connection, here are the steps needed:

Obtain a security certificate signed by a certificate authority. Doing this requires generating a signature request for the lighttpd web server on the PowerMedia XMS system. This is done by using the openssl utility on the PowerMedia XMS server:
> openssl req -new -nodes -keyout xms20.dialogic -out xms20.csr

The keyout parameter is the full domain name of the system on which PowerMedia XMS is installed. Questions will be asked as part of the signature request process. The resulting base 64 signature request text (.csr file) is then used when requesting the security certificate from the certificate authority.

The security certificate that is returned by the certificate authority must then be activated. This will involve loading the certificate into the existing cacerts keystore that comes with the Java JDK or JRE. This is done within the jre/lib/security directory with:
> keytool –import –alias hostname.domainname –file cert_from_authority.crt –keystore cacerts

This is intended to be a rough guide only. Follow the instructions provided by the certificate authority for using its security certificate with Java HTTPS applications.

To fully activate a legitimate security certificate, one further step is necessary. In the HTTPInterface class, there is an overridden method that has been added to disable hostname verification when a self-signed certificate is used for development/testing. This method needs to be ignored. This can be done with the "-l" or "—verify_hostname" option when running the demo.

The RESTful Management Demo

The demo code itself is well commented, so details of its operation will not be given here. However, an overview is provided below for assistance and to summarize what the demo does.

There are three parts to the demo that show how to manipulate Routes, Backups and Media files. In doing this, all of the HTTP methods and techniques needed for the full API are exercised. The JSON strings that are returned as response payloads are converted to JSON objects and reformatted to be displayed or sometimes modified with new data.

Routing Resources – PowerMedia XMS routes map SIP username/extension patterns to various application technologies such as RESTful, VXML and NetAnn. A table of routes is maintained and may be modified as desired by working with the routing resource. GETs and PUTs only are allowed, so the following procedure is used to add and then delete a new route:

1. Send a GET to retrieve all existing routes
2. Parse the JSON returned and add a new route to the list
3. Do a PUT to update the routing resource with the new set of routes
4. Use a PUT to add the original set of routes back, eliminating the one just added and returning the system to its original condition

System Backup Resource – a backup resource is a PowerMedia XMS system backup file, and contains the files needed to recreate a PowerMedia XMS installation. How HTTP methods are used in this context might not be apparent, and are illustrated by the following scenario:

1. Use GET to obtain a listing of all current backup files
2. Use a POST to instruct PowerMedia XMS to create a new system backup. The name of the new backup is returned in the response to the POST
3. Use a GET with a specific URI that points to the backup file just created to download the entire backup file as a binary payload as part of the response to the GET. The backup file is saved on the Java side
4. Use DELETE with a specific URI pointing to the just-downloaded backup file to delete it on the PowerMedia XMS system
5. Use GET to verify that the backup has been deleted and the system is in its original condition

Media Resource – media resources are all of the media files (audio, video and still image) on the PowerMedia XMS system. They may be listed, added and deleted using the various HTTP methods:

1. Send a GET to look at a listing of all of the media resources

2. DELETE a specific media file by using its full URI
3. The Java demo contains a copy of the file just deleted. Use a POST to upload the file to PowerMedia XMS
4. Verify that it is there with another GET

Running the RESTful Management Demo

The demo is delivered both as a Java JAR file and as a Java NetBeans project in a zipped file that should be unzipped before use. The NetBeans IDE version used with to develop the demo was 7.0.1, running on top of Java Development Kit (JDK) 1.6_017. A Java Runtime Environment (JRE) only is needed to run the JAR file.

Output from the demo is written to the console from which it is run and to a log file called RestMgmntDemo.log.

To run the JAR file from a command line, set the working directory to the top level XMSRestMgmnt directory and execute the JAR file in the "dist" directory:

```
> java –jar dist/ XMSRestMgmnt.jar <options>
```

Options are:

```
usage: XMSRestMgmnt [-h] -i <xms_ip_address> [-k <java_keystore_file>]  [-n] [-v]
 -h,--help                      Usage help
 -i,--ip_address <arg>          IP Address of PowerMedia XMS server
 -k,--keystore <arg>            Java Certificate Keystore file
 -l –verify_hostname            Do legitimate hostname verification
 -n,--network_logging           Log Java network layer
 -v,--verbose                   Do verbose application logging
```

Details on the options are given below:

--ip_address <address> is the IP address of the PowerMedia XMS server being managed

--keystore <keystore_file_location> Note that forward slashes are used, even for a Windows system – e.g. "C:/Program Files/ Java/jdk1.7.0_09/jre/lib/security"

--verify_hostname will not put in place a method to override hostname checking. This method is needed when a self-signed certificate is used.

--verbose will log DEBUG level information, INFO level otherwise. ERROR and FATAL will always be logged.

--network_logging will give a detailed log of HTTP. Very useful if network-related errors are encountered.

The project is ready to load into the NetBeans IDE. Use Open Project and select the top level project directory XMSRestMgmnt. The project may then be modified, built and run as desired.

Open Source

This technote discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Using the Dialogic® PowerMedia™ Extended Media Server (XMS) 2.0 RESTFul Management API - A Java Demo

Tech Note

Dialogic®

www.dialogic.com

Dialogic Inc
1504 McCarthy Boulevard
Milpitas, California 95035-7405
USA

NETWORK FUEL™