

Using XMLBeans to Simplify Dialogic®
PowerMedia™ Extended Media Server (XMS)
RESTful Application Programming in Java

Introduction

A Dialogic® PowerMedia™ Extended Media Server (XMS) application, as RESTful, web-based software, relies on an Extensible Markup Language (XML) payload to send and receive information to/from the server side. XML payloads are often defined using the XML Schema Document (XSD) formal schema description language. This is the case with PowerMedia XMS. Representational State Transfer (RESTful) payloads are described by an XSD document on the server side, and this same document may be used to generate XML language bindings for the client/application side. This article will describe one specific XSD processor, XMLBeans, and how it can be used as part of a Java PowerMedia XMS RESTful application, the PowerMedia XMS Verification Demo. These techniques can be applied to other PowerMedia XMS Java applications as well.

XMLBeans

Apache XMLBeans is a technology for accessing XML by binding it to Java types. Using XMLBeans, XML schema can be compiled to generate Java types that represent the schema. In this way, you can access instances of the schema through JavaBeans-style accessors after the fashion of "getFoo" and "setFoo". While a simple command line processor is used for creating the bindings and the Java JAR file, getting the options correct can be a hit or miss process. For this reason, scripts to run the processor (for both Windows and Linux) are included in this article.

Installation

A system with PowerMedia XMS installed is not necessary to create the xmsrest Java classes; only the latest xmsrest.xsd file from the XMS download is needed. It is located in the /etc/xms directory on an installed PowerMedia XMS system. See the [the Dialogic PowerMedia XMS Download](#) page to get started downloading, installing and configuring PowerMedia XMS.

Installing XMLBeans is covered in its README.txt file in its top level directory. However, a synopsis of what you need to do for processing an PowerMedia XMS schema is included here (release numbers current as of October 2012):

- Java - make sure you have Java Development Kit (JDK) 1.6.x installed, that java[.exe] is on your path and that the JAVA_HOME environment variable is set
- Download and uncompress the latest XMLBeans binary distribution, version 2.5.0. It may be found [here](#).
- Set the environment variable XMLBEANS_LIB to the installations lib directory. For example, in Linux, the .bash_profile file would have a line similar to "export XMLBEANS_LIB=/usr/java/xmlbeans-2.5.0/lib
- Copy a current xmsrest.xsd, from (/etc/xms on an installed XMS system) xmsrest.xsdconfig and the Windows (.bat) or Linux (.sh) script to xmlbeans-2.5.0 directory. All except for the current xmsres.xsd may be found at the end of this article.

Producing the JAR File

Simply run the script on the Windows or Linux command line:

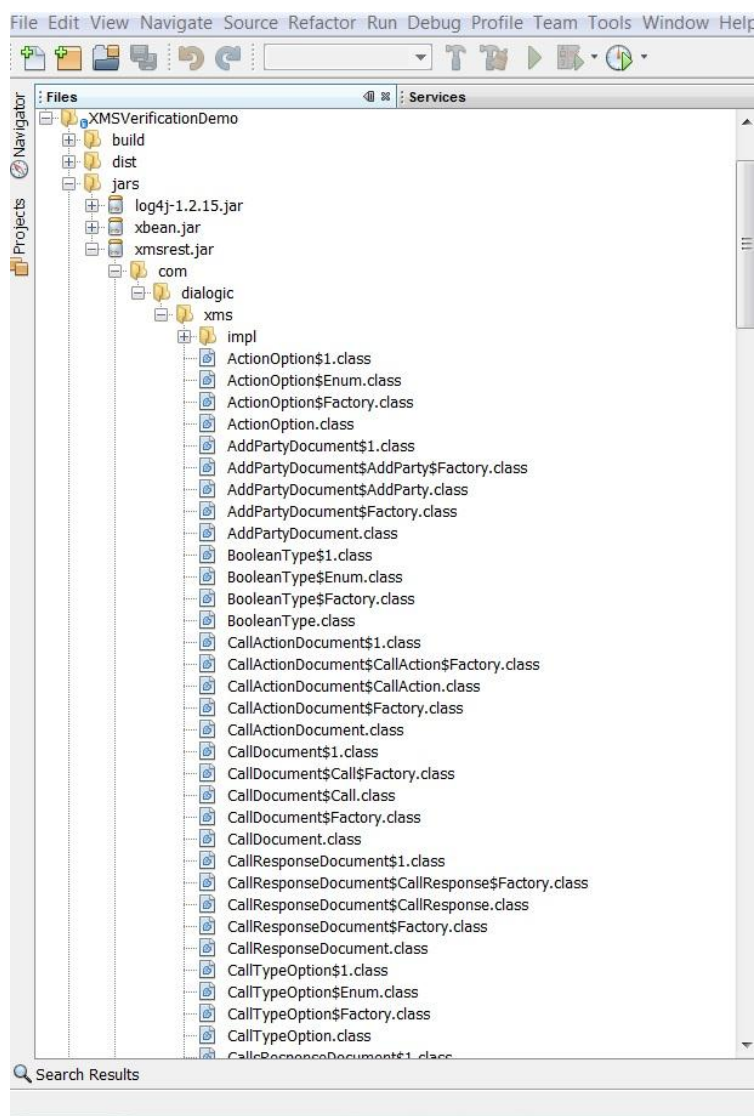
```
> make_jar[.bat or .sh]
```

Using XMLBeans to Simplify Dialogic® PowerMedia™ Extended Media Server (XMS) RESTful Application Programming in Java

Tech Note

A file by the name of xmsrest.jar will be produced. Copy this into your Java project under a jars or libs directory. In addition, there are dependencies on a number of classes in xbean.jar. This jar file is found under the "libs" directory of the XMLBeans distribution. It should also be copied into the same location as xmsrest.jar.

Further steps, depending on the IDE, will be needed to add the jar files to your build path. Here is the jar file embedded in the PowerMedia XMS Verification Demo, using the NetBeans IDE:



Using the Java Classes

Several examples of using the Java classes to build new XML requests and parsing out elements in responses will now be reviewed. Further generic examples can be found in the "samples" directory in the XMLBeans distribution.

Making an Outbound Call with PowerMedia XMS

An outbound call is done using an HTTP POST method. The HTTP document can be easily created and formatted. Then the XML payload can also be created and set up easily in terms of the actual numbers of lines of code and the actual formatting of the message itself:

```
// Create the HTTP document
WmsDoc = WebServiceDocument.Factory.newInstance();
WebServiceDocument.WebService wms= WmsDoc.addNewWebService();

XmlNMToken version = XmlNMToken.Factory.newInstance();
version.setStringValue("1.0");

// Create and setup a new call resource
Call newCall = wms.addNewCall();
newCall.setDestinationUri(toURI);
newCall.setCpa(BooleanType.NO);
newCall.setSdp("");
newCall.setMedia(MediaType.AUDIOVIDEO);
newCall.setSignaling(BooleanType.YES);
newCall.setDtmfMode(DtmfModeOption.RFC_2833);
newCall.setAsyncDtmf(BooleanType.YES);
newCall.setAsyncTone(BooleanType.YES);
newCall.setRxDelta("+0dB");
newCall.setTxDelta("+0dB");
newCall.setSourceUri(fromURI);
wms.xsetVersion(version);

// Convert the class to a string of XML for the HTTP request
XML=WmsDoc.toString();

// Send the HTTP request to media server and get the response
response = XMSInterface.SendHTTPRequestAndGetResponse(XMSInterface.getURL() + "calls",
"POST", XML);
```

Parsing a PlayRecord Response

Requests to play and record media will have a transaction ID for the media operation embedded in the response. This allows the play/record to be programmatically terminated. The transaction ID is needed to identify the operation and will usually be parsed out of the HTTP response when it is received. The reason that the operation terminated is also retrieved. Here is an example of a response to a PlayRecord request:


```
// Send the HTTP request to XMS
response = XMSInterface.SendHTTPRequestAndGetResponse(XMSInterface.getURL() +
fullPathOfPlayRecord, "PUT", XML);

WebServiceDocument wsDoc;
try {
    wsDoc = WebServiceDocument.Factory.parse(response) ;
    WebServiceDocument.WebService ws = wsDoc.getWebService();
    // Information on the result of the call action, including the transaction ID and
    // the reason the record terminated can now be parsed out
    CallResponse cR = ws.getCallResponse();
    CallAction cA = cR.getCallAction();
    Playrecord playRecordStatus = cA.getPlayrecord();
    String playTransactionID = playRecordStatus.getTransactionId();
} catch (XmlException ex) {
    logger.error(ex);
}
```

Links

The following are links regarding the major components used in this article:

- Dialogic® PowerMedia™ Extended Media Server (XMS) –
 - <http://www.dialogic.com/products/media-server-software/xms.aspx>
- XMLBeans - <http://xmlbeans.apache.org>
- NetBeans - <http://netbeans.org>
- Java - <http://www.java.com>

Open Source

This article 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.

Dialogic®

www.dialogic.com

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

Copyright © 2013 Dialogic Inc. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Inc. at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Inc. and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see <http://www.dialogic.com/company/terms-of-use.aspx> for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Inc. at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 6700 de la Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, BorderNet, PowerMedia, ControlSwitch, I-Gate, Mobile Experience Matters, Network Fuel, Video is the New Voice, Making Innovation Thrive, Diastar, Cantata, TruFax, SwitchKit, Eiconcard, NMS Communications, SIPcontrol, Exnet, EXS, Vision, inCloud9, NaturalAccess and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Inc. and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 6700 de la Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

03/13

NETWORK FUEL™