Tech Note

Dialogic® PowerMedia™ Extended Media Server (XMS) RESTful Web Applications Using Google Web Toolkit and the Atmosphere Project

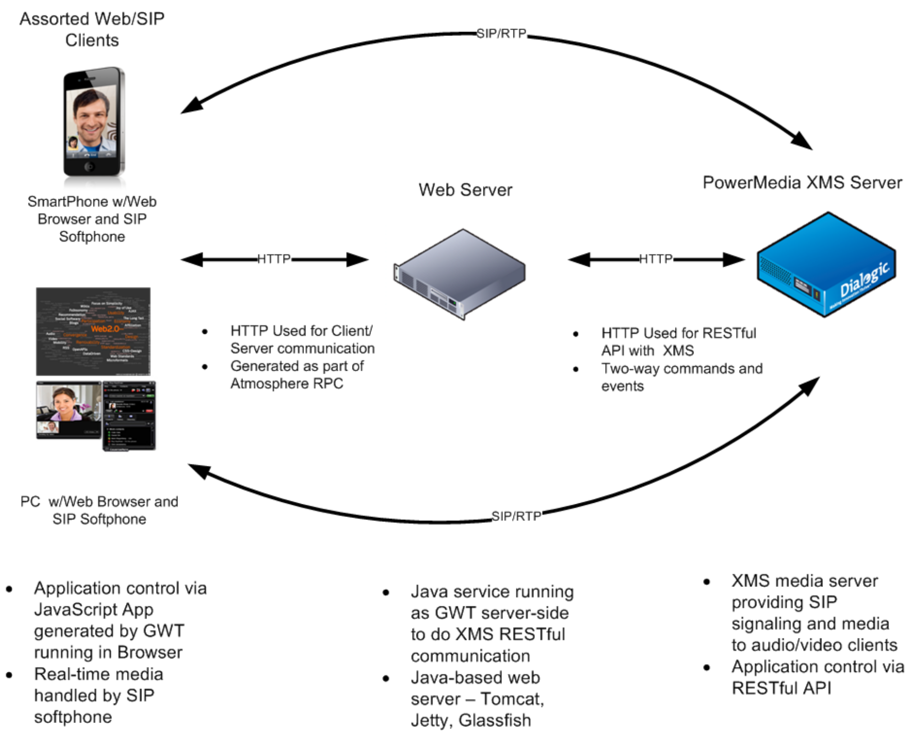
Introduction

Using a web-based user interface (UI) is a common way for users to interact with applications. This method can provide convenience and flexibility if the services used are located somewhere on the Internet and the client's system is used primarily as the user interface.

This model also can work well for many types of applications, particularly transaction-oriented ones. But there are special considerations when using real-time oriented add-ons like the Dialogic® PowerMedia™ Extended Media Server (PowerMedia XMS). Operations such as establishing calls, playing media, joining conferences often take tens of seconds, and the user cannot be left with a paused web browser while the operation concludes. The user may want to do other things on the page while waiting. In addition, it is important to report back the progress of the operation as it is happening. Finally, the user may choose to abort an operation after initiating it.

Full-duplex real-time communication between the web browser and the backend server is needed to accomplish this. This technote shows a way this sort of communication can be achieved. It uses the well-known Google Web Toolkit as a web application development environment. In addition, it uses the Atmosphere Framework, a set of client and server side components for building asynchronous web applications.

Overall Architecture

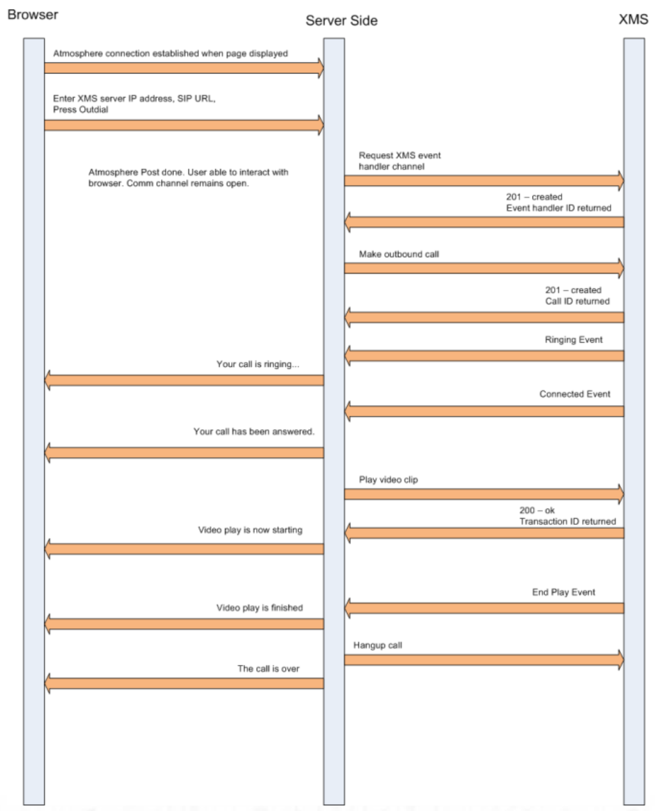

Assorted Web/SIP Clients
SIP/RTP
SmartPhone w/Web Browser and SIP Softphone
Web Server
PowerMedia XMS Server
HTTP
HTTP
HTTP Used for Client/ Server communication
Generated as part of Atmosphere RPC
HTTP Used for RESTful API with XMS
Two-way commands and events
PC w/Web Browser and SIP Softphone
SIP/RTP
Application control via JavaScript App generated by GWT running in Browser
Real-time media handled by SIP softphone
Java service running as GWT server-side to do XMS RESTful communication
Java-based web server – Tomcat, Jetty, Glassfish
XMS media server providing SIP signaling and media to audio/video clients
Application control via RESTful API

This RESTful web application uses a multi-tiered web application architecture with three distinct parts:

1. The client application downloaded to and running in the browser, communicating via HTTP with
2. A server application running in the web server. This, in turn, talks to
3. The PowerMedia XMS media server, accessed via an HTTP RESTful API

One possible method for achieving RESTful API control is to provide an HTTP connection between the web browser running a Javascript client and the PowerMedia XMS's RESTful interface. Using Javascript has limitations in that a media server needs to report back asynchronous events on activities such as making a call, playing a media file, or adding a party to a conference. Javascript and most client-side web browser programming languages are not multi-threaded, and if a single threaded browser app waited for a response, the user would have to wait as well. Some way of allowing real-time two-way communication between the browser on one side and handling asynchronous events from PowerMedia XMS on the other is required.

Multi-tiered architecture, with a Java-based server in the middle and two-way async communications on both sides is a solution.
On the browser/client side, the Atmosphere project is used to allow messages to be easily sent between the browser and the server-side application, written in Java, running on the web server. This server-side app is multithreaded, and its independent threads communicate with the client and with the PowerMedia XMS server as well. It relays commands and events of interest between the browser and PowerMedia XMS, and allows for filtering out unimportant event notifications or rewording events into something meaningful for the user. The following diagram shows the messaging between the various tiers for this demo application:


Browser
Server Side
XMS
Atmosphere connection established when page displayed
Enter XMS server IP address, SIP URL, Press Outdial
Request XMS event handler channel
Atmosphere Post done. User able to interact with browser. Comm channel remains open.
201 – created Event handler ID returned
Make outbound call
201 – created Call ID returned
Ringing Event
Your call is ringing...
Connected Event
Your call has been answered.
Play video clip
200 – ok Transaction ID returned
Video play is now starting
End Play Event
Video play is finished
Hangup call
The call is over

Between PowerMedia XMS and the Java server-side, a Comet web application model is used in PowerMedia XMS to achieve an open HTTP communications channel so that asynchronous events can be delivered to the web server and potentially relayed to the web browser client.

While a multi-tiered web application architecture is used to encapsulate application logic and for application control, a media and signaling connection using a SIP video phone is still needed to set up the call and receive/transmit the multimedia stream.

Components Used

- **Dialogic® PowerMedia™ Extended Media Server (PowerMedia XMS)** - is a powerful software media server that enables standards-based, real-time multimedia communications solutions for mobile and broadband environments. It offers a rich variety of advanced media processing functions, including audio and video play/record and content streaming, and can be used to build a wide variety of real-time multimedia processing solutions.
- **Google Web Toolkit -** Google Web Toolkit (GWT) is a development toolkit for building and optimizing complex browser-based applications, such as productive development of high-performance web applications without the developer generally having to be an expert in browser quirks, XMLHttpRequest, and JavaScript.
- **Atmosphere Project -** Atmosphere: The Asynchronous WebSocket/Comet Framework. Normalizes real-time communication between a variety of web browsers and back-end servers. Atmosphere may use Comet or use WebSockets, depending on the capabilities of the browser. However, the API presented to the application programmer does not need to take into account the communication method and browser used.
- **Apache Maven** - The Apache Maven build automation tool is used by Atmosphere to distribute its sample projects, so it is used with this demo. It provides a convenient development environment to debug, build, package and manage a project overall. The open source Jetty Java-based web server and servlet container is bundled into Maven's development environment. This provides a convenient with quick turnaround for application startup and test when a change is made to the code. Apache Maven version was 3.0.4.
- **Apache Tomcat** - Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. In other words - an open source web server written in Java. It was chosen as a web server for this demo as it is easy to install, configure and deploy a web application on it. Tomcat version was 7.0.29.
- **Eclipse -** The Eclipse IDE is used mainly for file management and ongoing compilation. While there are Eclipse plugins for GWT and Maven, it is generally easier to run the demo outside of Eclipse using Maven. The NetBeans IDE can also likely be used for the same purpose. The Eclipse version that was used was 3.6.1, and was automatically installed as part of CentOS 6.3 Development Tools.
- **Java -** Since development was being actively done, a Java Development Kit (JDK) was used instead of a Java Runtime Environment. (JRE) Java version was 1.6.0_24, automatically installed as part of CentOS 6.3 Development Tools. The equivalent JRE can also be used if the demo will only be run.
- **CentOS -** CentOS is an Enterprise-class Linux Distribution derived from sources freely provided to the public by Red Hat. Version 6.3 was used. A "Development Workstation" installation was chosen.

Installing and Configuring the Demo

The demo is delivered both as a Maven project and as a ready-to-use .war file. To simply run the project, installing Tomcat and deploying the .war file there is a quick and easy method. Detailed instructions for Tomcat installation can be found on the Apache-Tomcat website- http://tomcat.apache.org/tomcat-7.0-doc/appdev/installation.html

Deploying a .war file in Tomcat can be done in the Tomcat Manager screen accessed through a browser running on the same system as the .war file:

- First, use http://localhost:8080 as the URL
- For security reasons, it is necessary to add a user with manager capabilities to Tomcat after installation. See the section on Managing Tomcat on the initial startup page for instructions on how to do this
- Once a manager is added, use the URL http://localhost:8080/manager .Under DEPLOY/WAR file to deploy, browse to the .war file (GWTAtmosphereWebApp/Warfile/atmosphere-gwt-demo.war) and open it
- Hit the Deploy button. An application named atmosphere-gwt-demo will be made available
- Select the atmosphere-gwt-demo application to start it in this window or use another browser to access it at http://<ip_address>:8080/atmosphere-gwt-demo

Browsers Tested

Browser	**Version**	**Comments**
Firefox	3.6.9 Linux 64-bit	No observed issues
Firefox	10.0.4 Linux 64-bit	No observed issues
Firefox	14.0.1 Windows 64-bit	Stop Play command (2nd Atmosphere Post) did not appear until the Outbound Call (Original Atmosphere Post) was done. Incompatible with Google Web Toolkit's debug mode.
Internet Explorer	8.0.7600.16385 Windows 32-bit	No observed issues
Google Chrome	20.0.1132.57 m	Stop Play command (2nd Atmosphere Post) did not appear until the Outbound Call (Original Atmosphere Post) was done
Safari	5.1.7	Not able to get an Atmosphere connection
Opera	12.00	Not able to get an Atmosphere connection

Running the Demo

The demo consists of a single web page, which contains the controls to interact with PowerMedia XMS for running an outbound multimedia SIP call.   A SIP phone client will be needed to take the call.   There are free or pay-for SIP phone clients available. The SIP phone client must be able to take inbound calls and either auto-answer them or allow the user to click on Answer.

A PowerMedia XMS server should first be started and put into RESTful mode. See the Dialogic PowerMedia XMS Download page to get started downloading, installing and configuring PowerMedia XMS. Depending on the version of PowerMedia XMS used, the media files for the video clip may or may not be available. Copies of video_clip_nascar.vid and video_clip_nascar.wav are included in the media directory in the Atmosphere-gwt-demo project. If these files do not exist in the /var/lib/xms/media/en_US/verification directory on the PowerMedia XMS server, they should be copied there.

The next steps are:

- Bring up the demo web page. When the Tomcat web server is being used, the URL for the demo will be http://<web_server_ip_address>:8080/atmosphere-gwt-demo. Log output from the demo will be saved under <tomcat_install_dir>/logs/catalina.out. Note that the Tomcat server listens for web connections on port 8080. If CentOS's default firewall settings are used, port 8080 will be blocked. You can use the "Setup" tool enable TCP on port 8080, or, though not recommend, you also could turn off the firewall.
- On the demo web page, enter the IP address of the server under XMS Server IP Address
- Enter the SIP URL for the outbound call under SIP URL for Outdial. This is typically in the format sip:username@<phone_ip_address>
- Initiate the call. The SIP phone will ring and should be answered. On answer, a one minute video clip will be played to the caller. When the clip is over, the server hangs up the call.
- Two outdial options are available:
    - The first (GWT RPC) uses a simple GWT remote procedure call. This provides a single exchange of messages between the browser client and the server side that interacts with PowerMedia XMS. This is adequate for a simple outbound call, but does not provide interaction when the call is in progress. Results are reported when the call is complete. No other interaction is possible.
    - The second uses Atmosphere to provide a full-duplex communications channel for the duration of the call. Call setup and media events are reported back to the web browser as they happen. Sending a message to PowerMedia XMS is also possible. The Stop Play button will allow the user to stop the play and hang up the call when the play is in progress.

Modifying the Demo

While running the demo only requires the .war file, modifying the source and doing any further development needs a full development environment.

Full Development Environment

The following development environment was used on Linux CentOS 6.3, although other Java development environments are also possible. For modifications and further development of the demo, make sure that the following additional components are installed:

- **Java** – Java JDK 1.6.0_24, automatically installed as part of CentOS 6.3 Development Tools
- **Eclipse** - Eclipse was used to edit source files and compile them. Eclipse version was 3.6.1, automatically installed as part of CentOS 6.3 Development Tools. Two additional Eclipse installs are needed:
    - A "yum install" of Eclipse Plugin Development Environment, (PDE) eclipse-pde-3.6.1-6.13.el6.x86_64
    - A Maven plugin, which is needed to import, compile and edit a Maven project. In Eclipse, under "Help –> Install New Software -> Work with" add the repository "maven - http://download.eclipse.org/technology/m2e/releases" . Select "M2e - Maven Integration for Eclipse" and use the installation wizard to complete the installation.

  It is then possible to import the atmosphere-gwt-demo Maven project:

    1. Go to Open File -> Import -> Existing Maven Projects
    2. Continue with Next -> Browse. Got to the directory containing the Maven Project Object Model (POM) file for the demo, "samples/gwt-demo"
    3. Click OK and the project will be imported. It can then be opened, and the individual files examined and compiled with Eclispe. Note that "Build Automatically" should be enabled by default, and changes will be immediately compiled.
    4. Instructions for running the project in development mode are given in the next section.
- **Apache Maven**. Command-line Maven was used (rather than the Eclipse Maven plugin) for compiling and packaging the project. This is a separate Maven installation done by downloading Maven 3.0.4 zipped or tar file from Apache/Maven - http://download.eclipse.org/technology/m2e/releases Decompress it and install according to the instructions in README.txt. The download also contains the Google Web Toolkit components needed for the demo.

Editing, Compiling and Running the Project in Development Mode

As previously mentioned, Eclipse is used for editing and ongoing compilation of the demo. However, command-line Maven is used subsequent to that. To build and run the project:

```
> cd GWTAtmosphereWebApp/DevEnv/samples/gwt-demo
> mvn gwt:run
```

A Jetty web server is built into Maven, and will be used to serve up the web application. A window for the Jetty server appears, and also allows for a web browser to be started and the XMS demo page immediately accessed. Note that X Windows must be running at this point. The first time the demo is run in this manner, a maven plug-in will likely be required for Firefox on CentOS 6.3. Allow the plugin to be installed.

All demo application-level logging will appear in the console where the "mvn gwt:run" command is issued. The demo does not log to a file, so it may be desirable to run as:

```
> mvn gwt:run | tee gwt-demo.log
```

Packaging the PowerMedia XMS demo into a .war file for use with other web servers such as Tomcat is done with:

```
> mvn package
```

The .war file will be found under the "target" directory.

A full recompilation and repackaging can be done with:

```
> mvn clean install package
```

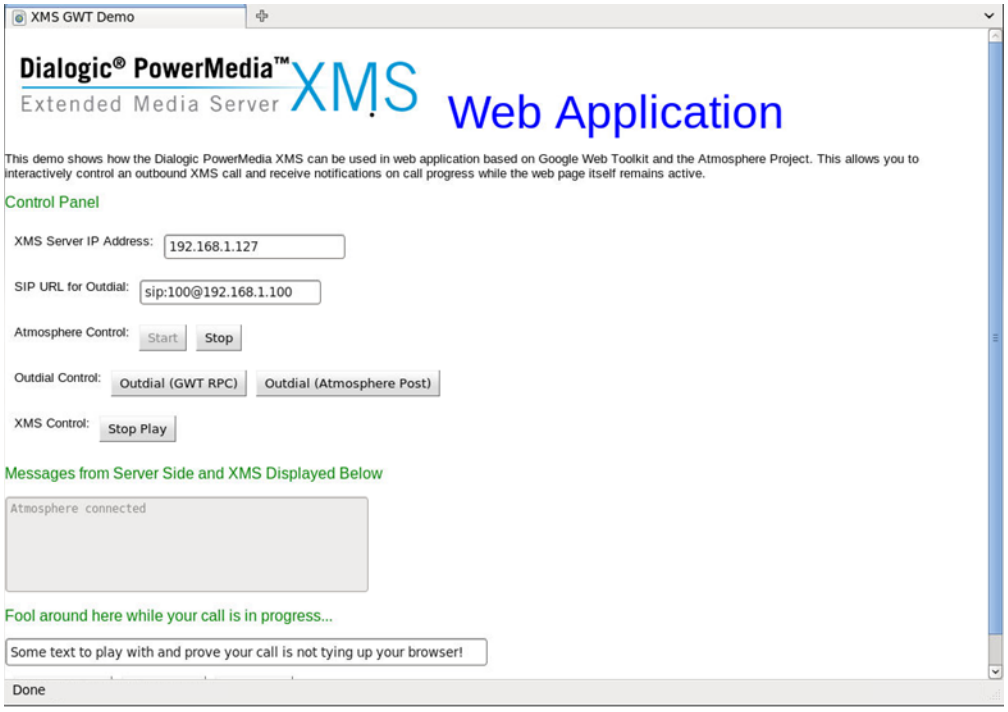
Internal Demo Description

As mentioned above, a multi-tiered web architecture is used for this demo. To recap the components:

1. A client side application, run on a web browser, coded in Java, and translated to downloadable JavaScript by GWT
2. An intermediate server, run on the web server, communicating with the web client on one side and PowerMedia XMS on the other. It is a multi-threaded application written in Java
3. The PowerMedia XMS server, providing the SIP signaling and media connections to the SIP client
4. The SIP phone/client, which may be co-located with the web browser

Client Side

This is a typical GUI-oriented application, with a variety of GWT UI widgets arranged on a web page.The placement, groupings and labels for the widgets are first given. This is best illustrated with the web page itself:



This demo shows how the Dialogic PowerMedia XMS can be used in web application based on Google Web Toolkit and the Atmosphere Project. This allows you to interactively control an outbound XMS call and receive notifications on call progress while the web page itself remains active.

Control Panel

XMS Server IP Address: 192.168.1.127

SIP URL for Outdial: sip:100@192.168.1.100

Atmosphere Control: Start Stop

Outdial Control: Outdial (GWT RPC) Outdial (Atmosphere Post)

XMS Control: Stop Play

Messages from Server Side and XMS Displayed Below

```
Atmosphere connected
```

Fool around here while your call is in progress...

Some text to play with and prove your call is not tying up your browser!

Done

The functional groups have unique names so that they may be interspersed with images and text in the web page definition, gwtDemo.jsp.

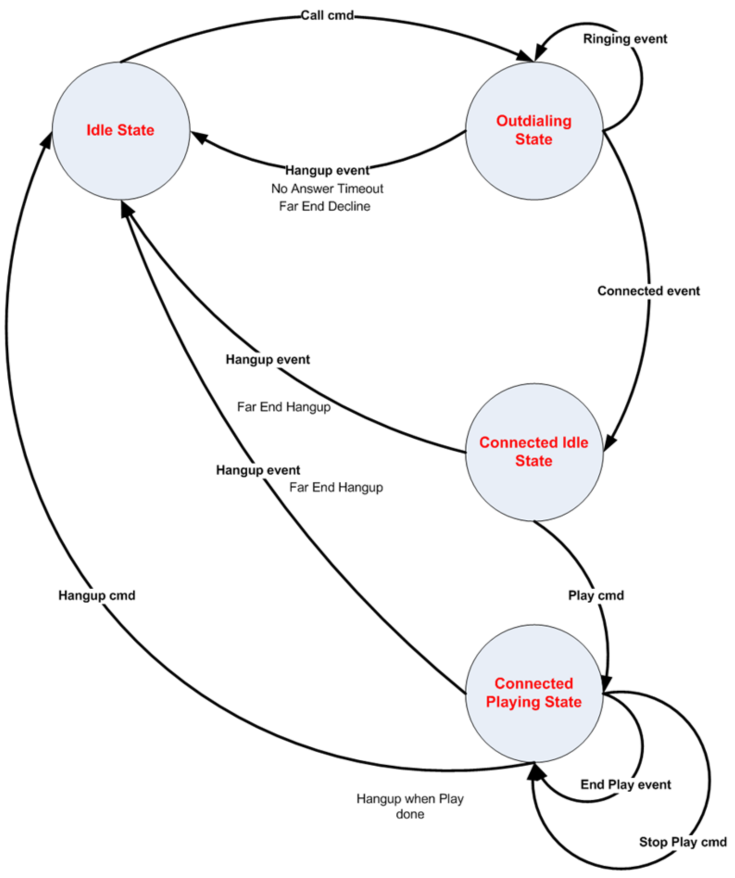
Code-wise, most important are the callback functions that are invoked when an action is taken by the user. This would include:

- Doing a simple GWT-based async request to PowerMedia XMS, via the server side, to trigger an outbound call and report back the results at the end of the call. This option is provided to show that while there is a standard GWT remote procedure call (RPC) mechanism available, it is not adequate for these purposes
- Doing an async connection to the server side using Atmosphere. The outbound call is triggered when connected, and a variety of intermediate results and progress messages are relayed back to the client side and displayed on in the scrolling text box on the web page
- A Stop Play button that shows how a message may be sent, via Atmosphere, that can be used asynchronously to control the media server during a call
- Several buttons that allow the user to play with a text box. This is to convince the user that the client side JavaScript app is functional while two-way Atmosphere communication is in progress with the server side.

Server Side

As mentioned, this is a multi-threaded Java application. One thread handles the Atmosphere communication with the web client and the other handles the communication with PowerMedia XMS. Commands are typically relayed from the web client and are translated into RESTful commands understood by the PowerMedia XMS RESTful API. At the same time, asynchronous events from PowerMedia XMS are read, typically translated into something more meaningful for the user, and sent to the web browser for display.

Business/call flow logic is also embedded into the server side. It is done using an object-oriented state machine:



Using an object-oriented state machine can be an efficient way of delineating the steps needed to process a multi-media call, and specifying the media-related events that make up the application – playing files, collecting menu choices, creating conferences, transferring calls, etc. Although the demo is rather simple – a single video clip is played and the call hung up – it still can benefit from being designed, coded and debugged using a clearly written state machine.

RESTful messages between the server side and PowerMedia XMS are built and parsed in a series of command objects. Code to send an HTTP request and receive a response is contained in a separate object. Building and parsing the HTTP XML payloads is done here.

However, a more generalized approach to doing this is described in this technote, (JH fix link) where a Java package called XMLBeans uses the PowerMedia XML schema to create Java classes for message building and parsing.

Finally, there are server-side objects for both Atmospehre and GWT async communication. Methods are invoked when a message from the client side is received. Methods are also available to asynchronously post a message to the client side (Atmosphere) and send the final GWT result from the server side.

Further Development

This demo is relatively simple – a single outbound call with a file played on answer. But it could make a good starting point for a reader interested in writing his/her own GWT application that uses PowerMedia XMS. As an example, a video conference control application would lend itself well to a web interface:

- A conference management GUI could provision and create a conference
- Emails could be sent to invitees to call in and join
- Or, outbound calls could be made directly to invitees
- Conference monitoring could be done through the web GUI – PowerMedia XMS can be easily polled for the number and types of conferees

As previously mentioned, a SIP phone is necessary for transmitting and receiving media. However, it could be eliminated through the use of a Flash-SIP gateway, making the application environment browser-based. A Flash Phone (usually implemented in JavaScript) is downloaded to the web browser from the gateway/web server. There, it is automatically invoked and establishes signaling and media paths back to the server and uses the PC's microphone, video camera and display to send/receive media.

While Google Web Toolkit was used as a basis for web development with this demo, there are different ways to go about web development. The Atmosphere Project – which is needed for the real-time communications needed with a media server - can be used with a wide variety of web development environments. The options are listed here.

Links

The following are links (as of March 2013) to major components used in this demo:
PowerMedia XMS – http://www.dialogic.com/products/media-server-software/xms.aspx
Google Web Toolkit – https://developers.google.com/web-toolkit/overview
Atmosphere Project – https://github.com/Atmosphere/atmosphere
Maven Project – http://maven.apache.org/
Tomcat Project – http://tomcat.apache.org/
Eclipse Project – http://www.eclipse.org/
Java - http://www.java.com

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.

www.dialogic.com

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