Application Notes for Packaging and Deploying Avaya Communications Process Manager Sample SDK Web Application on Apache Geronimo – Issue 1.0

Abstract

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager (CPM) sample Software Development Kit web application on an Apache Geronimo application server.

Apache Geronimo is an open source Java Enterprise Edition 5.0 application server. It supports web and Java 2 Enterprise Edition (J2EE) applications. For these Application Notes, a sample Avaya Communications Process Manager web client application available in the CPM SDK was deployed to Apache Geronimo. This client application was used to verify the Avaya CPM Simple Object Access Protocol (SOAP) web services interaction between the Avaya CPM and Apache Geronimo.
1. Introduction

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager (CPM) sample SDK web application on an Apache Geronimo application server.

Apache Geronimo is an open source Java Enterprise Edition 5.0 application server. It supports web and Java 2 Enterprise Edition (J2EE) applications. For these Application Notes, a sample Avaya CPM web client application available in the Avaya CPM SDK was recompiled and deployed to Apache Geronimo. This web client application was used to verify the CPM Simple Object Access Protocol (SOAP) Web services interaction between the Avaya CPM and Apache Geronimo. The CPM sample web client application project was written in Java code and is generic to all platforms. These Application Notes illustrate how to recompile and deploy this sample CPM web client application to Apache Geronimo.

The Avaya CPM SDK sample web client application supports the following Avaya Communication Enabled Business Process web services:

- **Advisory**: This web client application initiates an outbound advisory request to a list of recipients for them to acknowledge receipt of the notification.

- **Notify and Respond**: This web client application initiates an outbound notification with a set of questions to a list of recipients and waits for them to respond to the notification.

- **Notify and Conference**: This web client application initiates an outbound notification to a list of recipients. When a notified user answers the phone, the service provides contextual information about the exception conference and asks if the caller wants to join the conference.

- **Find and Call**: This web client application uses a recipient list to create either a two-party call or an on demand conference.

For detailed information on the sample Avaya SDK web client application refer to [1] in the Additional References section. For these Application Notes, the Avaya CPM SDK web client application was used to verify the interoperability between Apache Geronimo and the Avaya CPM via web services.
2. Configuration

Figure 1 provides an overview of the network used in the sample configuration. This sample configuration consists of Avaya S8710 Servers with an Avaya G650 Media Gateway running Avaya Communication Manager, Avaya SIP Enablement Services (SES), Avaya Voice Portal, Nuance Text to Speech server, Avaya Meeting Exchange Express, Avaya Communications Process Manager and Apache Geronimo. The Avaya 9620 IP Telephone (H.323) and Avaya 9620 IP Telephone (SIP) are registered with Avaya Communication Manager and Avaya SES, respectively. Avaya SIP telephones are configured as Off-PBX stations (OPS). Avaya Voice Portal serves as an interactive voice response system for converting CPM text messages to voice.
3. Equipment and Software Validated

The following equipment and software were used for the sample configuration provided:

<table>
<thead>
<tr>
<th>Equipment &amp; Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avaya S8710 Servers</td>
<td>Avaya Communication Manager 5.0.1 (R015x.00.0.825.4)</td>
</tr>
<tr>
<td>Avaya G650 Media Gateway</td>
<td></td>
</tr>
<tr>
<td>TN2302AP MEDPRO</td>
<td>HW20 FW116</td>
</tr>
<tr>
<td>TN799DP CLAN</td>
<td>HW01 FW024</td>
</tr>
<tr>
<td>Avaya Meeting Exchange Express</td>
<td>2.5.60.0</td>
</tr>
<tr>
<td>Avaya SIP Enablement Services</td>
<td>5.0.0.0.0-825.31</td>
</tr>
<tr>
<td>Avaya Voice Portal</td>
<td></td>
</tr>
<tr>
<td>• VPMS</td>
<td>4.0.0.0.2901</td>
</tr>
<tr>
<td>• MPP</td>
<td></td>
</tr>
<tr>
<td>Nuance TTS server</td>
<td></td>
</tr>
<tr>
<td>Real speak</td>
<td>4.0.10</td>
</tr>
<tr>
<td>Avaya 9620 IP Telephone</td>
<td>2.0.1.34 (SIP)</td>
</tr>
<tr>
<td>Avaya 9620 IP Telephone</td>
<td>1.5 (H.323)</td>
</tr>
<tr>
<td>Avaya Communications Process Manager</td>
<td>Release 2.1</td>
</tr>
<tr>
<td>Avaya Communications Process Manager SDK</td>
<td>Release 2.1</td>
</tr>
<tr>
<td>Apache Geronimo</td>
<td></td>
</tr>
<tr>
<td>• Microsoft Windows 2003 Server</td>
<td>Release 2.1</td>
</tr>
<tr>
<td>• Sun Microsystems Java JDK</td>
<td>Service Pack 2</td>
</tr>
<tr>
<td></td>
<td>1.5_0_14</td>
</tr>
<tr>
<td>Developer PC Windows XP Professional</td>
<td>2002 SP 2</td>
</tr>
<tr>
<td>• Sun Microsystems Java JDK</td>
<td>1.5_14</td>
</tr>
<tr>
<td>• Eclipse</td>
<td>3.3.1.1</td>
</tr>
<tr>
<td>• Geronimo Eclipse Plugin</td>
<td>v2.0.0</td>
</tr>
<tr>
<td>Microsoft Active Directory and DNS Server</td>
<td></td>
</tr>
<tr>
<td>Microsoft Windows Server 2003</td>
<td>Service Pack 2</td>
</tr>
</tbody>
</table>

Table 1: Equipment/Software List
4. Configure Avaya Communications Process Manager

In these Application Notes, it is assumed that the Avaya CPM software and the license file have already been previously installed. These Application Notes further assume that Avaya Communication Manager, Avaya SES, Avaya Voice Portal, and Avaya Meeting Exchange Express (as shown in Fig. 1) have already been configured and are operational with Avaya CPM. For additional information on these installation tasks, refer to [3] – [6] in the Additional References section. This section describes the steps that are required for adding users to the Avaya CPM for use with the sample SDK web application described in later sections.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>On the Avaya CPM server, launch a web browser and enter the URL http://&lt;Name or IP address of CPM server&gt;/VIA. When prompted for a user name and password, enter the credentials of the administrator account.</td>
</tr>
<tr>
<td>2.</td>
<td>The Communications Process Manager Administration Portal page appears. To add a new user, click Account.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>3.</td>
<td>The <strong>Account Home</strong> page appears. To create a new user account, for the CPM Notifications and Response system, click <strong>Create Account</strong>.</td>
</tr>
</tbody>
</table>

### Account Home
- **Account Information**
  - Specify personal information including your time zone
  - Change your PIN
  - Specify your preferences for being contacted
- **Points Of Contact**
- **Time Profiles**
- **Notification Profiles**
- **Notification Filters**

### Account creation
You can search for users by entering letters or digits in the search box in the sidebar. If you then click on **Search**, or press **Enter**, the system will return those users whose handle (user id) or common name matches the sequence of letters, or whose phone number contain the given digits. (The match is actually more liberal, if the handle or common name contains the digits, they will match as well.) You may also enter a [LDAP Search Filter](#) - All `inetOrgPerson` attributes are supported.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4.   | The **Create Account** screen appears. The **Create Account** screen is shown in **Steps 5 and 6**. Enter the following values:  
  - **Administrator**: To give the user administrator access, select **Yes**.  
  - **CPM User**: To make this user a licensed Communications Process Manager user, select **Yes**. Licensed Communications Process Manager Users can log in to the Communications Process Manager portal and receive notifications.  
  - **Handle**: Enter the user’s handle (e.g., **Bill**).  
  - **ID Number**: A unique ID number (e.g., **39101**) for the user. Can be an employee number or something similar. This ID number serves as the user's account number and initial PIN.  
  - **Last Name**: Enter User's last name (e.g., **Smith**).  
  - **Common Name(s)**: Communications Process Manager typically uses the first value of the common name in interactions with the user.  
  - **Phone Number**: Telephone number at which Communications Process Manager contacts the user (e.g., **sip:39101@cebp-avaya.com**). This is the user telephone number administered in SES server (not shown in these Application Notes).  
  - **Advisory Service**: Select **yes**.  
  - **Click To Find Service**: Select **yes**.  
  - **Notification And Response Service**: Select **yes**.  
  - **Notify And Conference Service**: Select **yes**.  
  
  Click **Save**. |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td></td>
</tr>
</tbody>
</table>

**Roles**

- **Administrator:** ☐ Yes ☑ No
- **CPM User:** ☐ Yes ☑ No

**Attributes**

- **Handle:** Bill
- **ID Number:** 39101
- **Display Name:** 
- **First Name:** 
- **Last Name:** Smith
- **Common Name(s):** Bill
- **Phone Number:** sip:39101@cebp-avaya.com
- **Mobile Phone Number:** 
- **Fax Number:** 
- **Pager Number:** 
- **Electronic Mail Address:** 
- **Honorific:** Mr.
- **Title:** 
- **Affiliation:** ☐ Employee
e
- **Manager:** 

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td></td>
</tr>
</tbody>
</table>

| Department: |          |
| Organization: |          |
| Room: |          |
| Street: |          |
| City: |          |
| State: |          |
| Postal Code: |          |
| Language: | US English |
| Time Zone: |          |
| Applications |          |

- **Advisory Service**: Yes  No
- **Click To Find Service**: Yes  No
- **Notification And Response Service**: Yes  No
- **Notify And Conference Service**: Yes  No

7. Repeat **Step 4** to create two additional CPM user accounts (i.e., Jenny and Jimmy).
5. Compile Sample Avaya Communications Process Manager
Web Client Application

This section illustrates how to compile the Avaya CPM SDK web client application on a
developer PC running on Windows XP. These Application Notes assume that the Eclipse
Geronimo plugin is already installed on the developer PC. Refer to [8] in the Additional
References section for detailed installation information. The CPM Client SDK can be
downloaded from the following URL: http://www.avaya.com/devconnect. For more
information on the sample Avaya SDK web client application refer to [7] in the Additional
References section.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>From the developer PC launch Eclipse. The Eclipse page appears.</td>
</tr>
</tbody>
</table>
2. To import the CPM clientsdk-webui project, navigate to **File → Import.** The **Import page** appears. Click **Existing Projects into WorkSpace.** Click **Next.**
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>In the <strong>Select root directory</strong> field, enter the root directory of the CPM clientsdk-webui project (i.e., “C:\CPMSDK2.1\javasdk\clientsdk-webui”). The <strong>clientsdk-webui</strong> project appears in the <strong>Projects</strong> pane. Click <strong>Finish</strong>.</td>
</tr>
</tbody>
</table>

![Import dialog box](image)
### Step 4

Verify that the **clientsdk-webui** project appears in the **Project Explorer** pane.

**Note:** The red “x” next to **clientsdk-webui** indicates the clientsdk-webui project has not been successfully built. Steps 5 – 6 will clear the error indication.
5. To modify the validation preferences, right click on `clientsdk-webui` and then click Properties. The Properties for clientsdk-webui page appears. Click on Validation. Uncheck HTML Syntax Validator and JSP Syntax Validator in both the Manual and Build columns. Click OK.

Note: If validation is not required, then check Disable all to disable all validations.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>To add CPM_LIB as a class path variable, right click on <code>clientsdk-webui</code> and then click <strong>Build Path → Configure Build Path</strong>. <strong>Java Build Path</strong> appears in the right pane. Click <strong>Add Variable → Configure Variables → New</strong> to add the “CPM_LIB” variable pointing to the “C:/CPMSDK-2.1/javasdk/lib” folder. Click <strong>OK</strong>.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>7.</td>
<td>Verify that the error indication shown on Step 4 is cleared.</td>
</tr>
<tr>
<td>8.</td>
<td>To create a CPM web client war file, from the <strong>Project Explorer</strong> pane, click on <strong>build.xml</strong>, and then select <strong>Run → Run</strong> from the file menu. The <strong>Run As</strong> window appears. Click <strong>Ant Build</strong> and then click <strong>OK</strong>.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>9.</td>
<td>When the build is completed successfully, the <strong>CPMClient-2.1.war</strong> file is created under the <strong>dist</strong> folder. The result of the Ant build is shown in the screen below.</td>
</tr>
</tbody>
</table>
6. Deploy the Sample Avaya Communications Process Manager Client Application to Apache Geronimo

This section illustrates how to deploy the CPM SDK client application WAR file\(^1\), created in Section 5 Step 9, to Apache Geronimo. These Application Notes assume that Apache Geronimo is already installed on a Microsoft Windows 2003 server. Refer to [8] in the Additional References section for the installation. These Application Notes further assume that a proper version of the Sun JDK is downloaded and configured.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Launch a web browser and enter the URL <code>http://&lt;Name or IP address of Apache Geronimo Server&gt;:8080/console</code>. When prompted for a user name and password, enter the credentials of the Apache Geronimo Tomcat administrator account.</td>
</tr>
<tr>
<td>2.</td>
<td>In the <strong>Apache Geronimo</strong> screen that appears, click <strong>Deploy New Applications</strong>.</td>
</tr>
</tbody>
</table>

\(^1\) Alternatively, the WAR file shipped with the Avaya CPM SDK can be used to deploy to the Apache Geronimo server.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>The <strong>Install New Applications</strong> widow appears. Enter the absolute path and file name of the WAR, created in <strong>Section 5, Step 9</strong>, (i.e., “C:\CPMSDK2.1\javasdk\clientsdk-webui\dist\CPMClient-2.1.war”). Click <strong>Install</strong>.</td>
</tr>
</tbody>
</table>
7. Verification Steps

The following steps may be used to verify proper configuration of Avaya CPM and Apache Geronimo:

- Verify that the sample Avaya CPM Web Client application is deployed to Apache Geronimo.
- Verify that the CPM Advisory service can be launched and acknowledged by all the recipients.
- Verify that CPM Notify and Conference service can be launched and the recipients can join the conference.
- Verify that CPM Notify and Respond service can be launched and the recipients can respond to the notification.
- Verify that CPM Find and Call service can be launched and conference the recipients.
- Verify Avaya CPM & Apache Geronimo log files do not show any errors.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1.   | Verify that the CPMClient web application can be started from Apache Geronimo.  
  - Launch a web browser, enter the URL [http://<IP address of Apache Geronimo>:8080/CPMClient-2.1](http://<IP address of Apache Geronimo>:8080/CPMClient-2.1).  
  - Verify that the **Communications Process Manager Client** screen appears. |

![Communications Process Manager Client](AVAYA_CommunicationsProcessManagerClient.png)

**CPM Services**

- [Advisory Service](#)
- [Notify and Conference](#)
- [Notify and Respond](#)
- [Find and Call](#)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Verify that the CPM Advisory Service can be launched by clicking <strong>Advisory Service</strong>. Fill in the fields with appropriate values. Verify that the recipients acknowledge receipt of the notification.</td>
</tr>
<tr>
<td>3.</td>
<td>Repeat Step 2 and verify that CPM Notify and Conference service can be launched. Verify that the Notify and Conference service is successfully completed.</td>
</tr>
<tr>
<td>4.</td>
<td>Repeat Step 2 and verify that CPM Notify and Respond service can be launched. Verify that the Notify and Respond service is successfully completed.</td>
</tr>
<tr>
<td>5.</td>
<td>Repeat Step 2 and verify that CPM Find and Call service can be launched. Verify that the Find and Call service is successfully completed.</td>
</tr>
</tbody>
</table>
6. Launch a web browser and enter the URL http://<Name or IP address of CPM server>/VIA to view the debug screen as necessary for verifying the web client application execution. When prompted for a user name and password, enter the credentials of the CPM user account. The Communications Process Manager Inbox screen appears. Click Outbox. Select the following fields to view the completed notifications.

- Select “Completed” from the Select field to view the completed notifications.

The completed notifications are displayed as shown below.
To view the contents of a notification, click the subject of the notification.

<table>
<thead>
<tr>
<th>Application</th>
<th>Session Id</th>
<th>Start Date</th>
<th>End Date</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Click To Find Service</td>
<td>AAAAAKvwXpc=KdZ0y/y==</td>
<td>Thursday 21 February 2008, 3:18:22 pm</td>
<td>Thursday 21 February 2008, 3:19:37 pm</td>
<td>Invitation to conference call.</td>
</tr>
<tr>
<td>2 Notification &amp; Response Service</td>
<td>AAAAAKvwXpy=HdZnig==</td>
<td>Thursday 21 February 2008, 3:17:01 pm</td>
<td>Thursday 21 February 2008, 3:17:40 pm</td>
<td>Vacation plan</td>
</tr>
<tr>
<td>3 Notify And Conference Service</td>
<td>AAAAAKvwXpu=HdZnSg==</td>
<td>Thursday 21 February 2008, 3:13:59 pm</td>
<td>Thursday 21 February 2008, 3:14:54 pm</td>
<td>Geronimo</td>
</tr>
<tr>
<td>4 Advisory Service</td>
<td>AAAAAKvwXpQ=KdZm2ge=</td>
<td>Thursday 21 February 2008, 3:04:17 pm</td>
<td>Thursday 21 February 2008, 3:04:56 pm</td>
<td>CPM with Geronimo</td>
</tr>
</tbody>
</table>

Delete Selected Request(s)
8. Troubleshooting Steps

This section contains simple troubleshooting steps for debugging purposes.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The following log files are available on the Avaya CPM for troubleshooting purposes. The log files are stored under the <code>/var/log/cpm</code> directory.</td>
</tr>
<tr>
<td></td>
<td>- mail.log</td>
</tr>
<tr>
<td></td>
<td>- nafssvc.log</td>
</tr>
<tr>
<td></td>
<td>- oam.log</td>
</tr>
<tr>
<td></td>
<td>- dcore.log</td>
</tr>
<tr>
<td></td>
<td>- commflow.log</td>
</tr>
<tr>
<td></td>
<td>- advsvc.log</td>
</tr>
<tr>
<td></td>
<td>- b2bua.log</td>
</tr>
<tr>
<td></td>
<td>- account.log</td>
</tr>
<tr>
<td></td>
<td>- via.log</td>
</tr>
<tr>
<td></td>
<td>- platform-licensing.log</td>
</tr>
<tr>
<td></td>
<td>- user-licensing.log</td>
</tr>
<tr>
<td></td>
<td>- cs-tomcat-memorymonitor.log</td>
</tr>
<tr>
<td></td>
<td>- cpm.log</td>
</tr>
<tr>
<td></td>
<td>- complete.log</td>
</tr>
</tbody>
</table>
2. To view the Geronimo log entries, from the Apache Geronimo server console, click **Server Logs**. Expand the **Log Viewer** (for example, **Server Log Viewer**) to view the log files.
9. Conclusion

As illustrated in these Application Notes, Apache Geronimo can be used to successfully interoperate with Avaya Communications Process Manager via web services.

10. Additional References

The following document may be obtained from http://support.avaya.com.


The Apache Geronimo documents may be obtained from

11. Appendix A – build.xml file

This section shows the build.xml file, (used in Section 5, Step 8), as a reference on how to build the sample SDK web application into CPMClient-2.1.war file.

<?xml version="1.0" encoding="UTF-8"?>
<!--
General purpose build script for web applications and web services, including enhanced support for deploying directly to a Tomcat 5 based server.

This build script assumes that the source code of your web application is organized into the following subdirectories underneath the source code directory from which you execute the build script:

docs       Static documentation files to be copied to the "docs" subdirectory of your distribution.

src        Java source code (and associated resource files) to be compiled to the "WEB-INF/classes" subdirectory of your web applicaiton.

web        Static HTML, JSP, and other content (such as image files), including the WEB-INF subdirectory and its configuration file contents.

$Id: build.xml.txt 302898 2004-05-23 19:50:44Z markt $
-->

<!-- A "project" describes a set of targets that may be requested when Ant is executed. The "default" attribute defines the target which is executed if no specific target is requested, and the "basedir" attribute defines the current working directory from which Ant executes the requested task. This is normally set to the current working directory. -->

<project name="CPMClientSdkWUI" default="dist" basedir=".">

<!-- ==================== File and Directory Names ======================== -->

<!-- These properties generally define file and directory names (or paths) that affect where the build process stores its outputs. -->

app.name       Base name of this application, used to construct filenames and directories. Defaults to "myapp".

app.path       Context path to which this application should be deployed (defaults to "/" plus the value of the "app.name" property).

app.version    Version number of this iteration of the application.

build.home     The directory into which the "prepare" and
“compile” targets will generate their output. Defaults to "build".

catalina.home    The directory in which you have installed a binary distribution of Tomcat 5. This will be used by the "deploy" target.

dist.home       The name of the base directory in which distribution files are created. Defaults to "dist".

-->

<property name="app.name" value="CPMClient"/>
<property name="app.path" value="/${app.name}"/>
<property name="build.home" value="${basedir}/build"/>
<property name="catalina.home" value="../../../.."/>
<property name="dist.home" value="${basedir}/dist"/>
<property name="docs.home" value="${basedir}/docs"/>
<property name="src.home" value="${basedir}/src"/>
<property name="src.lib" value="${basedir}/..lib"/>
<property name="web.home" value="${basedir}/WebContent"/>
<property name="app.version" value="2.1"/>

<!-- ============== Compilation Control Options ============== -->

<!-- These properties control option settings on the Javac compiler when it is invoked using the <javac> task. -->

compile.debug    Should compilation include the debug option?
compile.deprecation Should compilation include the deprecation option?
compile.optimize  Should compilation include the optimize option?

-->

<property name="compile.debug" value="true"/>
<property name="compile.deprecation" value="false"/>
<property name="compile.optimize" value="true"/>

<!-- ============== External Dependencies ============== -->

<!-- Use property values to define the locations of external JAR files on which your application will depend. In general, these values will be used for two purposes: -->

* Inclusion on the classpath that is passed to the Javac compiler
* Being copied into the "/WEB-INF/lib" directory during execution of the "deploy" target.

Because we will automatically include all of the Java classes that Tomcat 5 exposes to web applications, we will not need to explicitly list any of those dependencies. You only need to worry about external dependencies for JAR files that you are going to include inside your "/WEB-INF/lib" directory.

-->
<property name="foo.jar" value="/path/to/foo.jar"/>

<!--
Rather than relying on the CLASSPATH environment variable, Ant includes features that makes it easy to dynamically construct the classpath you need for each compilation. The example below constructs the compile classpath to include the servlet.jar file, as well as the other components that Tomcat makes available to web applications automatically, plus anything that you explicitly added.
-->

<path id="compile.classpath">
  <!-- Include all JAR files that will be included in /WEB-INF/lib -->
  <!-- *** CUSTOMIZE HERE AS REQUIRED BY YOUR APPLICATION *** -->
  <!--
  <pathelement location="${foo.jar}"/>
  -->
  <!-- Include all elements that Tomcat exposes to applications -->
  <!--
 <pathset dir="${catalina.home}/common/classes"/>
  <include name="*.jar"/>
  </pathset>
  <fileset dir="${catalina.home}/common/endorsed">
    <include name="*.jar"/>
  </fileset>
  <fileset dir="${catalina.home}/common/lib">
    <include name="*.jar"/>
  </fileset>
  <pathelement location="${catalina.home}/shared/classes"/>
  <fileset dir="${catalina.home}/shared/lib">
    <include name="*.jar"/>
  </fileset>
  <fileset dir="${src.lib}">
    <include name="**/*.jar"/>
  </fileset>
  </pathset>
  </path>

<!--
The "prepare" target is used to create the "build" destination directory, and copy the static contents of your web application to it. If you need to copy static files from external dependencies, you can customize the contents of this task.

Normally, this task is executed indirectly when needed.
-->
<mkdir dir="${build.home}/WEB-INF/classes"/>
<mkdir dir="${dist.home}"/>

<!-- Generate the service proxy -->
<ant antfile="../clientsdk-serviceproxy/build.xml" inheritAll="false"
target="dist"/>

<!-- Copy static content of this web application -->
<copy todir="${build.home}"
     fileset dir="${web.home}"/>
</copy>

<!-- Copy external dependencies as required -->
<!-- *** CUSTOMIZE HERE AS REQUIRED BY YOUR APPLICATION *** -->
<mkdir dir="${build.home}/WEB-INF/lib"/>

<!-- Copy external dependencies as required -->
<!-- *** CUSTOMIZE HERE AS REQUIRED BY YOUR APPLICATION *** -->
<copy todir="${build.home}/WEB-INF/lib" file="${foo.jar}"/>

<!-- Copy static files from external dependencies as needed -->
<!-- *** CUSTOMIZE HERE AS REQUIRED BY YOUR APPLICATION *** -->
</target>

<!-- ==================== Clean Target ================================== -->
<!--
The "clean" target deletes any previous "build" and "dist" directory, so that you can be ensured the application can be built from scratch.
-->
<target name="clean"
     description="Delete old build and dist directories">
<delete dir="${build.home}"/>
<delete dir="${dist.home}"/>
</target>

<!-- ==================== Compile Target =============================== -->
<!--
The "compile" target transforms source files (from your "src" directory) into object files in the appropriate location in the build directory. This example assumes that you will be including your classes in an unpacked directory hierarchy under "/WEB-INF/classes".
-->

<target name="compile" depends="prepare" description="Compile Java sources">

<!-- Compile Java classes as necessary -->
<mkdir dir="${build.home}/WEB-INF/classes"/>
<javac srcdir="${src.home}"
    destdir="${build.home}/WEB-INF/classes"
    debug="${compile.debug}"
    deprecation="${compile.deprecation}"
    optimize="${compile.optimize}"
    classpath refid="compile.classpath"/>
</javac>

<!-- Copy application resources -->
<copy todir="${build.home}/WEB-INF/classes">
    <fileset dir="${src.home}" excludes="**/*.java"/>
</copy>
</target>

<!-- ==================== All Target ====================================== -->

<!-- The "all" target is a shortcut for running the "clean" target followed
by the "compile" target, to force a complete recompile. -->

<target name="all" depends="clean,compile" description="Clean build and dist directories, then compile"/>

<!-- ==================== Dist Target ===================================== -->

<!-- The "dist" target creates a binary distribution of your application
in a directory structure ready to be archived in a tar.gz or zip file.
Note that this target depends on two others:
* "compile" so that the entire web application (including external
dependencies) will have been assembled
* "javadoc" so that the application Javadocs will have been created -->

<target name="dist" depends="compile, javadoc" description="Create binary distribution">

<!-- Copy documentation subdirectories -->
<mkdir dir="${dist.home}/docs"/>
<copy todir="${dist.home}/docs">
    <fileset dir="${docs.home}"/>
</copy>-->

<jar jarfile="${dist.home}/${app.name}-${app.version}.war" basedir="${build.home}"/>

<!-- Copy additional files to ${dist.home} as necessary -->
</target>
The "src-dist" target creates a source distribution of your application in a directory structure ready to be archived in a tar.gz or zip file. Note that this target depends on two others:

* "compile" so that the entire web application (including external dependencies) will have been assembled

<!-- The "src-dist" target creates a source distribution of your application in a directory structure ready to be archived in a tar.gz or zip file. Note that this target depends on two others:
* "compile" so that the entire web application (including external dependencies) will have been assembled

<target name="src-dist" depends="compile"
    description="Create source distribution">
    <zip destfile="${dist.home}/${app.name}-${app.version}-src.zip"
        update="true">
        <fileset dir="${basedir}"
            include="**/*"/
            exclude="ant-build/**"/
            exclude=".settings/**"/
            exclude="build/**"/
            exclude="dist/**"/
            exclude="**/.svn/**"/>
    </zip>
</target>