Abstract

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager (CPM) sample SDK web application on a BEA Weblogic Application Server (AS) using BEA Weblogic development tools and Apache Ant.

The BEA Weblogic AS is an open integration and application platform. It supports Java technology and provides Java Enterprise Edition web services. For these Application Notes, a sample Avaya CPM web client application available in the CPM SDK was deployed to the BEA Weblogic AS, using BEA Weblogic tools and Apache Ant. This client application was used to verify the Avaya CPM Simple Object Access Protocol (SOAP) web services interaction between the Avaya CPM and BEA Weblogic AS.
1. Introduction

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager (CPM) sample SDK web application on a BEA Weblogic Application Server (AS), using BEA Weblogic development tools and Apache Ant.

BEA Weblogic is an application server for use with Java 2 Platform, Enterprise Edition (J2EE) and Enterprise Java Beans (EJB). For these Application Notes, a sample Avaya CPM web client application available in the Avaya CPM SDK was recompiled, and the resulting Web Application Archive (WAR) file was deployed to the BEA Weblogic AS. This web client application was used to verify the CPM Simple Object Access Protocol (SOAP) Web services interaction between the Avaya CPM and the BEA Weblogic AS platform. The CPM sample web client application was written in Java and is platform independent. These Application Notes illustrate how to recompile and deploy this sample CPM web client application to the BEA Weblogic AS using Apache Ant.

The Avaya CPM SDK sample web client application supports the following Avaya CPM web services:

- **Advisory**: This web service initiates an outbound advisory request to a list of recipients for them to acknowledge receipt of the notification.
- **Notify and Respond**: This web service 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 service 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 [8] in the Additional References section.

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 Express, Avaya Communications Process Manager and the BEA Weblogic AS. The Avaya 4600 Series IP Telephones (both H.323 and SIP) are registered with Avaya Communication Manager (running on the Avaya S8710 Server) and Avaya SIP Enablement Services (SES), respectively. Avaya SIP telephones were configured as Off-PBX stations (OPS). Avaya Voice Portal serves as an interactive voice response system for converting CPM text message to voice.
3. Equipment and Software Validated

The following equipment and software were used for the sample configuration described in these Application Notes:

<table>
<thead>
<tr>
<th>Equipment &amp; Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avaya S8710 Server</td>
<td>Avaya Communication Manager 4.0.1 (R014x.00.1.731.2)</td>
</tr>
<tr>
<td>Avaya G650 Media Gateway</td>
<td></td>
</tr>
<tr>
<td>Avaya 4600 Series IP Telephones</td>
<td>2.2.2 (SIP)</td>
</tr>
<tr>
<td>Avaya 4600 Series IP Telephones</td>
<td>2.8 (H.323)</td>
</tr>
<tr>
<td>Avaya Communication Process Manager</td>
<td>R2.1.53</td>
</tr>
<tr>
<td>Avaya SIP Enablement Services</td>
<td>4.0-04.0.033.6</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>Avaya Meeting Exchange Express</td>
<td>2.5.60.0</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 Communications Process Manager SDK</td>
<td>2.1.53</td>
</tr>
<tr>
<td>BEA Weblogic Application Server</td>
<td>Version 10.0 (Build id: 933139)</td>
</tr>
<tr>
<td>• Microsoft Windows 2003 Server</td>
<td>Service Pack 2</td>
</tr>
<tr>
<td>• Apache Ant</td>
<td>1.6.5</td>
</tr>
<tr>
<td>• BEA JRockit jvm</td>
<td>build R26.4.0-63_CR302700-72606-1.5.0_06-20061127-1108-win-ia32</td>
</tr>
<tr>
<td>Microsoft Active Directory and DNS Server</td>
<td>Service Pack 2</td>
</tr>
<tr>
<td>• Microsoft Windows 2003 Server</td>
<td></td>
</tr>
</tbody>
</table>

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

These Application Notes assume that the Avaya CPM software and the license file have already been previously installed. It is further assumed 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 [1], [2], [3], [4], and [5] in Section 11.

This section describes the steps that are required for adding users to the Avaya Communications Process Manager 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 <code>http://&lt; CPM server Name or IP address &gt;/VIA</code>. When prompted for a user name and password, enter the credentials of the server administrator account.</td>
</tr>
<tr>
<td>2.</td>
<td>The Communication Process Manager Administration web portal page opens. To add a new user, click Account.</td>
</tr>
</tbody>
</table>
3. The **Account Home** page opens. To create a new user account, click **Create Account**.
The Create Account page opens. The Create Account screen is shown in Steps 5 and 6. Enter the following values to add a new user to the CPM LDAP directory.

- **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 E-mail handle (e.g., billsmith).
- **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 default telephone number at which Communications Process Manager contacts the user.
- **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>5. Roles</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator:</td>
<td>☑ Yes ☐ No</td>
</tr>
<tr>
<td>CPM User:</td>
<td>☑ Yes ☐ No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*Handle:</td>
<td>Bill</td>
</tr>
<tr>
<td>*ID Number:</td>
<td>39101</td>
</tr>
<tr>
<td>Display Name:</td>
<td></td>
</tr>
<tr>
<td>First Name:</td>
<td></td>
</tr>
<tr>
<td>*Last Name:</td>
<td>Smith</td>
</tr>
<tr>
<td>*Common Name(s):</td>
<td>Bill</td>
</tr>
<tr>
<td>Phone Number:</td>
<td>sip:<a href="mailto:39101@cebp-avaya.com">39101@cebp-avaya.com</a></td>
</tr>
<tr>
<td>Mobile Phone Number:</td>
<td></td>
</tr>
<tr>
<td>Fax Number:</td>
<td></td>
</tr>
<tr>
<td>Pager Number:</td>
<td></td>
</tr>
<tr>
<td>Electronic Mail Address:</td>
<td></td>
</tr>
<tr>
<td>Honorific:</td>
<td>Mr.</td>
</tr>
<tr>
<td>Title:</td>
<td></td>
</tr>
<tr>
<td>Affiliation:</td>
<td>☐ Employee</td>
</tr>
<tr>
<td>Manager:</td>
<td></td>
</tr>
<tr>
<td><strong>Applications</strong></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Advisory Service:</strong></td>
<td></td>
</tr>
<tr>
<td>☐ Yes ☐ No</td>
<td></td>
</tr>
<tr>
<td><strong>Click To Find Service:</strong></td>
<td></td>
</tr>
<tr>
<td>☐ Yes ☐ No</td>
<td></td>
</tr>
<tr>
<td><strong>Notification And Response Service:</strong></td>
<td></td>
</tr>
<tr>
<td>☐ Yes ☐ No</td>
<td></td>
</tr>
<tr>
<td><strong>Notify And Conference Service:</strong></td>
<td></td>
</tr>
<tr>
<td>☐ Yes ☐ No</td>
<td></td>
</tr>
</tbody>
</table>
5. Set up the BEA Weblogic Build Environment

This subsection configures the development workstation in preparation for compiling and deploying the CPM web client application to a BEA Weblogic AS. These Application Notes assume that BEA Weblogic and BEA Workshop including Apache Ant already are installed on the development workstation and they are operational. It is also assumed that the BEA Weblogic AS is installed, operational and available for deploying applications on the server. For additional information on BEA Weblogic and BEA Workshop, refer to [9] – [14] in the Additional References section.

These Applications Notes use:

- `c:\bea\wlserver_10.0\` as the `wl_home` directory. This is the BEA Weblogic installation directory.
- `wl_home\samples\domains\wl_server\` as the `domain_home` directory. This is the BEA Weblogic domain home. That is, this is the directory where the application server domain instance is located.
- `c:\bea\Avaya\CPMWebApp\CPM_BEA_ANT` as the `cpm_app` directory. The `cpm_app` directory contains the CPM client web application and library directories.

In the following discussions the reader needs to substitutes the appropriate path for the above path variable.

The CPM Client SDK is available for the registered Devconnect members. It can be downloaded from the following URL: [http://www.avaya.com/devconnect](http://www.avaya.com/devconnect). For more information on the sample Avaya SDK web client application, refer to [7] in the Additional References section.

Note: The BEA Weblogic development environment incorporates capabilities to start, reboot, shutdown, or connect to a Weblogic server instance using Ant tasks, if the server is not running during development. Run the `wl_home\server\bin\setWLSEnv.cmd` command to set the development environment for using these Ant tasks. Then use `wlserver` and `wlconfig` Ant tasks to perform the desired tasks. Refer to [12] in section 11 for additional information on these Ant tasks.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Create the <code>cpm_app</code> project directory for the CPM sample web application on the BEA Weblogic server.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.</td>
<td>Run the <code>domain_home\bin\setDomainEnv.cmd</code> command to set the domain environment for using Apache Ant with BEA Weblogic extensions. The domain home will change, if a different BEA Weblogic domain and server instance is used. Verify that the MS Windows PATH now includes the BEA Weblogic environment, as shown below. The <code>setDomainEnv.cmd</code> command will change the current directory. Therefore, change directory to the CPM web client application directory when finished.</td>
</tr>
</tbody>
</table>

```
C:\default_path >
C:\default_path> domain_home\bin\setDomainEnv.cmd
domain_home>
domain_home> PATH
PATH=C:\bea\patch_wls1000\profiles\default\native;C:\bea\patch_wlw1000\profiles\default\native;C:\bea\WLSE~1.0\server\native\win\32;C:\bea\WLSE~1.0\server\bin;C:\bea\modules\ORGAPA~1.5\bin;C:\bea\JROCKI~1\jre\bin;C:\bea\JROCKI~1\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WIN~1.0\server\native\win\32;oci920_8
C:\default_path>
cdm p_m_app
```

| 3.   | Copy the CPM SDK zip file onto the workstation where BEA Weblogic is installed. Expand the zip file, and note the location of the expanded directory. Copy the `clientsdk-webui` and `lib` directories from the expanded CPM SDK directory to the `cpm_app` directory. Change to the `cpm_app\clientsdk-webui` directory for the remaining steps. |

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
6. Compile Sample Avaya Communications Process Manager Web Client Application

This section presents the steps to compile the CPM Web client application using Apache Ant. BEA provides numerous Ant extension classes to help you compile, build, deploy, and package applications using the BEA Weblogic Server development environment.

The steps described in this section include:

- Creating a weblogic.xml file, to resolve the issue with BEA Weblogic version 10.0.
- Creating a Weblogic template build.xml file, as a guide for modifying the build.xml file.
- Modifying the CPM client web application build.xml file, in order to compile the application in a BEA Weblogic development environment.
- Compiling the CPM client web application using the modified build.xml file.

Note: In the following discussion the reader needs to substitute the appropriate path for the cpm_app path variable, as described above.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1.   | In BEA Weblogic 10.0, the `context.getRealPath()` method fails to return the CPM-Client-Keystore, when WAR files are deployed. This results in the web application failing with a nullpointer error upon deployment. A `weblogic.xml` file is created to resolve this issue.  

Create a new file named `weblogic.xml`, and save it in the `cpm-app\clientsdk-webui\WebContent\WEB-INF` directory. Open this file for editing, and add the text in the following screenshot to this file. Save and exit the `weblogic.xml` file. 

The `container-descriptor` lines in the `weblogic.xml` file resolves the bug described above. **\* This text is copy and pasted from the screenshot. **|

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90">
    <session-descriptor>
        <timeout-secs>3600</timeout-secs>
        <invalidation-interval-secs>60</invalidation-interval-secs>
        <persistent-store-type>memory</persistent-store-type>
        <url-rewriting-enabled>true</url-rewriting-enabled>
    </session-descriptor>
    <jsp-descriptor>
        <keepgenerated>true</keepgenerated>
        <page-check-seconds>5</page-check-seconds>
        <precompile>false</precompile>
    </jsp-descriptor>
    <container-descriptor>
        <show-archived-real-path-enabled>true</show-archived-real-path-enabled>
    </container-descriptor>
</weblogic-web-app>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2.   | The **build.xml** file that comes with the CPM SDK web client application needs to be modified for it to work with BEA Weblogic. BEA Weblogic provides a tool to generate a template for the **build.xml** file.  
Change directory to the `cpm-app\clientsdk-webui` directory. Run the command `java weblogic.BuildXMLGen <source directory> -file <filename>` in the `clientsdk-webui` directory, to create a BEA Weblogic **build.xml** template file. Use the `-file` option on the command, and name the file something other than **build.xml** so as not to overwrite the CPM client web application **build.xml** file.  
In the following screenshot the source directory is **src** and the template file created is **build-weblogic.xml**. Verify that the **build-weblogic.xml** template file was created. |
| 3.   | The reader can either modify the CPM client web application **build.xml** file or modify the BEA Weblogic template **build.xml** file in the `cpm_app\clientsdk-webui` directory. If doing the former the Ant tasks need to be updated with the appropriate BEA Weblogic Ant tasks. If doing the latter, the appropriate information from the CPM client web application **build.xml** file needs to be added to the BEA Weblogic build.xml template file. For these Application Notes, the CPM client web application **build.xml** file was modified.  
Open the **build.xml** file for editing. Copy the following build properties to the **build.xml** file, from the BEA Weblogic template **build.xml** file (i.e., **build-weblogic.xml**). |

```xml
<property name="ear" value="${dist.dir}/${app.name}.ear"/>
<property name="verbose" value="true" />
<property name="user" value="USERNAME" />
<property name="password" value="PASSWORD" />
<property name="servername" value="myserver" />
<property name="adminurl" value="iiop://localhost:7001" />
```
### Step Description

4. Modify the build properties as described below.

- Delete the `catalina.home` and `manager.url` build properties from the `build.xml` file.
- Replace the `ear` property name with `war`. Also, replace the property value with `${dist.home}/${app.name}-${app-version}.war`.
- Add a `debug` property with the value set to `true`.
- Replace the `user` build property value with the username of the BEA Weblogic Server.
- Replace the `password` build property value with the password of the BEA Weblogic Server.
- Replace the `servername` build property value with the BEA Weblogic server name. For these application notes it was the `exampleServer`.

```xml
<property name="war" value="${dist.home}/${app.name}-${app-version}.war"/>
<property name="debug" value="true"/>
<property name="verbose" value="true"/>
<property name="user" value="WLS_USERNAME"/>
<property name="password" value="WLS_PASSWORD"/>
<property name="servername" value="exampleServer"/>
<property name="adminurl" value="iiop://localhost:7001"/>
```

5. Delete the **Compilation Control Options** section of the `build.xml` file, since these controls aren’t used by the BEA Weblogic compiler. That is, delete the following lines from the `build.xml` file.

```xml
<!-- =============== 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"/>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>Delete the <strong>catalina fileset</strong> lines from the <strong>Compilation Classpath</strong> section of the <strong>build.xml</strong> file, since the BEA Weblogic AS is being used. That is, delete the following lines from the <strong>build.xml</strong> file.</td>
</tr>
</tbody>
</table>

```xml
<!-- Include all elements that Tomcat exposes to applications -->
<fileset dir="${catalina.home}/common/classes"/>
<include name="*.jar"/>
</fileset>
<fileset dir="${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>
</include>
```

7. Delete the **Custom Ant Task Definitions** section of the **build.xml**. Also, delete all the related targets. That is, delete the following sections of the **build.xml** file.

- **Custom Ant Task Definitions**,  
- **Install Target**,  
- **List Target**,  
- **Reload Target**, and  
- **Remove Target**.

These Ant tasks are used to interact with the “/manger” application installed with Apache Tomcat 5. Equivalent BEA Weblogic Ant tasks will be added to the **build.xml** file later in this section.

8. The **clean** target deletes the build and distribution directories. There are no changes to the **clean** target.

Run the **clean** target and verify that the **dist** and **build** directories are deleted.

```bash
cpm-app\clientsdk-webui> ant clean
Buildfile: build.xml

clean:
[delete] Deleting directory cpm-app\clientsdk-webui\build
[delete] Deleting directory cpm-app\clientsdk-webui\dist

BUILD SUCCESSFUL
Total time: 5 seconds
C:\bea\Avaya\CPMWebApp\CPM_BEA_Ant\clientsdk-webui>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.</td>
<td>The <strong>prepare</strong> target creates the build directory and copies the static contents of the web application to it. There are no changes to the <strong>prepare</strong> target. Run the <strong>prepare</strong> target and verify that the prepare tasks succeed. The following screenshot shows the output from running the <strong>prepare</strong> target.</td>
</tr>
</tbody>
</table>

```
cpm-app\clientsdk-webui >
cpm-app\clientsdk-webui > ant prepare
Buildfile: build.xml
prepare:  
  [mkdir] Created dir: cpm-app\clientsdk-webui\build  
  [mkdir] Created dir: cpm-app\clientsdk-webui\build\WEB-INF  
  [mkdir] Created dir: cpm-app\clientsdk-webui\build\WEB-INF\classes  
  [mkdir] Created dir: cpm-app\clientsdk-webui\dist  
  [copy] Copying 15 files to cpm-app\clientsdk-webui\build  
  [copy] Copied 3 empty directories to 1 empty directory under cpm-app\clientsdk-webui\build  
  [copy] Copying 51 files to cpm-app\clientsdk-webui\build\WEB-INF\lib  
BUILD SUCCESSFUL  
Total time: 2 seconds  
cpm-app\clientsdk-webui >
```
10. The **compile** target builds the web application object files and places them in the correct directories. The **compile** target depends upon the **prepare** target. Modify the **compile** target by substituting the **wlcompile** the task for the **javac** task, and delete the **deprecation** and **optimize** parameters.

The **wlcompile** Ant task creates an **APP-INF/classes** directory tree in the **../WEB-INF/classes/** directory for the java classes. In addition, it creates a **../WEB-INF/classes/ com/** directory tree for the **ApplicationResources.properties** file. Therefore, the existing **copy** Ant task must be replaced with new **copy** and **delete** Ant tasks used create the correct directory and file structure for the CPM client web application.

The following screenshot shows the **compile** target used for these Application Notes.

```xml
<target name="compile" depends="prepare" description="Compile Java sources">
    <!-- Compile Java classes as necessary -->
    <mkdir dir="${build.home}/WEB-INF/classes"/>
    <wlcompile srcdir="${src.home}"
        destdir="${build.home}/WEB-INF/classes" >
        debug = ${debug}
        <classpath refid="compile.classpath"/>
    </wlcompile>

    <!-- Copy application resources -->
    <copy todir="${build.home}/WEB-INF/classes/com">
        <fileset dir="${build.home}/APP-INF/classes/com"/>
    </copy>
    <delete dir="${build.home}/APP-INF" />

    <copy todir="${build.home}/WEB-INF/classes/com/avaya/cpm/clientsdk/wui"
        file="${src.dir}/com/avaya/cpm/clientsdk/wui/ApplicationResources.properties"/>
</target>
```
11. Run the **compile** target and verify that the compile succeeds. The following screenshot shows the output from running the **compile** target.

```
cpm-app\clientsdk-webui>
cpm-app\clientsdk-webui>ant compile
Buildfile: build.xml
prepare:
  [mkdir] Created dir: cpm-app\clientsdk-webui\build
  [mkdir] Created dir: cpm-app\clientsdk-webui\build\WEB-INF
  [mkdir] Created dir: cpm-app\clientsdk-webui\build\WEB-INF\classes
  [mkdir] Created dir: cpm-app\clientsdk-webui\dist
  [copy] Copying 15 files to cpm-app\clientsdk-webui\build
  [copy] Copied 3 empty directories to 1 empty directory under cpm-app\clientsdk-webui\build
  [copy] Copying 51 files to cpm-app\clientsdk-webui\build\WEB-INF\lib
compile:
  [javac] Compiling 9 source files to cpm-app\clientsdk-webui\build\WEB-INF\APP-INF\classes
  [copy] Copying 9 files to cpm-app\clientsdk-webui\build\WEB-INF\classes\com
  [delete] Deleting directory cpm-app\clientsdk-webui\build\WEB-INF\classes\APP-INF
  [copy] Copying 1 file to cpm-app\clientsdk-webui\build\WEB-INF\classes\com\avaya\cpm\clientsdk\wui
BUILD SUCCESSFUL
Total time: 24 seconds
cpm-app\clientsdk-webui>
```
7. Create Sample Avaya CPM Web Client Application WAR File

This section describes the steps used to compile the (Java Server Pages) JSPs and generate a WAR file containing the CPM Web Client Application using Ant functions.

The steps described in this section include:

- Creating the WAR file,
- Compiling the JSPs, and
- Modifying the All Target to perform a complete recompile.

Note: In the following discussion the reader needs to substitute the appropriate path for the `cpm_app` path variable.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1.   | The **dist** target creates a standard J2EE WAR for distribution of the web application. That is, it creates the WAR file. The dist target depends upon the **compile** and **javadoc** targets.  

Open the build.xml file, and modify the **dist** target by replacing the **jar** Ant task with the **wlpackage** Ant task. The **wlpackage** Ant task packages a standard J2EE web application archive (WAR) for deployments. The lines that are replaced are shown in the following screenshot. |

```
<target name="dist" depends="compile, javadoc"

  
  Delete the following 2 lines

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

  Add the following 2 lines

  <wlpackage srcdir="${src.home}" destdir="${build.home}" toFile="${war}" />

```
2. **Step Description**

Run the `dist` target and verify that the war file is created. The following screenshot shows the output from running the `dist` target.

```plaintext
Buildfile: build.xml

prepare:
[mkdir] Created dir: cpm-app\clientsdk-webui\build
[mkdir] Created dir: cpm-app\clientsdk-webui\build\WEB-INF
[mkdir] Created dir: cpm-app\clientsdk-webui\build\WEB-INF\classes
[mkdir] Created dir: cpm-app\clientsdk-webui\dist
[copy] Copying 15 files to cpm-app\clientsdk-webui\build
[copy] Copied 3 empty directories to 1 empty directory under cpm-app\clientsdk-webui\build
[copy] Copying 51 files to cpm-app\clientsdk-webui\build\WEB-INF\lib

compile:
[javac] Compiling 9 source files to cpm-app\clientsdk-webui\build\WEB-INF\classes\APP-INF\classes
[copy] Copying 9 files to cpm-app\clientsdk-webui\build\WEB-INF\classes\com
[delete] Deleting directory cpm-app\clientsdk-webui\build\WEB-INF\classes\APP-INF
[copy] Copying 1 file to cpm-app\clientsdk-webui\build\WEB-INF\classes\com\avaya\cpm\clientsdk\wui

javadoc:
[mkdir] Created dir: cpm-app\clientsdk-webui\dist\docs\api
[javadoc] Generating Javadoc
[javadoc] Javadoc execution
[javadoc] Loading source files for package
com.avaya.cpm.clientsdk.wui...
[javadoc] Constructing Javadoc information...
[javadoc] Standard Doclet version 1.5.0_06
[javadoc] Building tree for all the packages and classes...
[javadoc] cpm-app\clientsdk-webui\src\com\avaya\cpm\clientsdk\wui\FindAndCallAction.java:159: warning
- @return tag has no arguments.
[javadoc] cpm-app\clientsdk-webui\src\com\avaya\cpm\clientsdk\wui\FindAndCallAction.java:337: warning
- @return tag has no arguments.
[javadoc] cpm-app\clientsdk-webui\src\com\avaya\cpm\clientsdk\wui\NotifyAndConferenceAction.java:434: warning
- @return tag has no arguments.
[javadoc] cpm-app\clientsdk-webui\src\com\avaya\cpm\clientsdk\wui\NotifyAndRespondAction.java:173: warning
- @return tag has no arguments.
[javadoc] Building index for all the packages and classes...
[javadoc] Building index for all classes...
[javadoc] Generating cpm-app\clientsdk-webui\dist\docs\api\stylesheet.css...
[javadoc] 4 warnings

dist:
[jar] Building jar: C:\bea\Avaya\CPMWebApp\CPM_BEA_Ant\clientsdk-webui\dist\CPMClient-2.1.war

BUILD SUCCESSFUL
Total time: 10 seconds
```
3. Create a new target named `appc` target that includes the `wlappc` Ant task. The `appc` target depends upon the `dist` target.

The BEA `weblogic.appc` compiler generates Java Server Pages (JSPs) and container-specific EJB classes for deployment, and validates deployment descriptors for compliance with the current Java EE specifications. `appc` performs validation checks between the application-level deployment descriptors and the individual modules in the application, as well as validation checks across the modules.

Add the text shown in the following screenshot to the `build.xml` for the `appc` target.

Note: The `appc` target must be run after the `dist` target is run.

```xml
<target name="appc" depends="prepare, compile, dist" description="Runs weblogic.appc on your application">
  <wlappc source="${war}" forceGeneration="true" debug="${debug}" verbose="${verbose}"
    <classpath refid="compile.classpath"/>
  </wlappc>
</target>
```

4. Run the `appc` target and verify that the Weblogic appc compiler generates the JSPs. The following screenshot shows the output from running the `appc` target.

```bash
cpm-app\clientsdk-webui >
cpm-app\clientsdk-webui > ant appc
Buildfile: build.xml
prepare:
  ●
  ● output the same as Step 9, Section 6
  ●
compile:
  ●
  ● output the same as Step 11, Section 6
  ●
javadoc:
  ●
  ● output the same as Step 2, Section 7
  ●
dist:
  [jar] Building jar: cpm-app\clientsdk-webui\dist\CPMClient-2.1.war
(continued)
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>Continuation of <strong>appc</strong> target output.</td>
</tr>
</tbody>
</table>

```plaintext
appc:
  [wlappc] Created working directory:
  C:\...\Temp\appcgen_1205507108749_CPMClient-2.1.war
  [wlappc] [JspcInvoker] Checking web app for compliance.
  <[ComplianceChecker] Validating the servlet element with servlet-name named "action".>
  <[ComplianceChecker] Validating the servlet element with servlet-name named "AxisServlet".>
  <[ComplianceChecker] Checking servlet-mapping for servlet name : "action".>
  <[ComplianceChecker] Checking servlet-mapping for servlet name : "AxisServlet".>
  <[ComplianceChecker] Checking servlet-mapping for servlet name : "AdminServlet".>
  <[ComplianceChecker] Checking servlet-mapping for servlet name : "AxisServlet".>
  <[ComplianceChecker] Checking servlet-mapping for servlet name : "AxisServlet".>
  <[ComplianceChecker] Checking servlet-mapping for servlet name : "AdminServlet".>
  [wlappc] [jspc] -webapp specified, searching . for JSPs
  [wlappc] [jspc] Compiling /Advisory.jsp
  [wlappc] [jspc] Compiling /FindAndCall.jsp
  [wlappc] [jspc] Compiling /index.jsp
  [wlappc] [jspc] Compiling /MonitorNotification.jsp
  [wlappc] [jspc] Compiling /NotifyAndConference.jsp
  [wlappc] [jspc] Compiling /NotifyAndRespond.jsp
  [wlappc] [jspc] Compiling /jsp_common/header.jsp
  [wlappc] Compilation completed successfully.
BUILD SUCCESSFUL
Total time: 4 minutes 54 seconds
cpm-app\clientsdk-webui >
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6.   | Modify the `all` target to replace the `compile` dependency with the `appc` dependency, (not shown). With this change the `all` target is a shortcut to running the `clean`, `compile`, `dist`, and `appc` targets. Thus forcing a complete rebuilding the CPM client web application war file in preparation for deployment.  

Run the `all` target and verify that the `clean`, `compile`, `dist`, and `appc` targets run and complete successfully, in this order. The following screenshot shows the output from running the `appc` target. |

```bash
cpm-app\clientsdk-webui > ant all
Buildfile: build.xml

clean:
  [delete] Deleting directory cpm-app\clientsdk-webui\build
  [delete] Deleting directory cpm-app\clientsdk-webui\dist
prepare:
  •
  • output the same as Step 9, Section 6
  •
compile:
  •
  • output the same as Step 11, Section 6
  •
javadoc:
  •
  • output the same as Step 2, Section 7
  •
dist:
  [jar] Building jar: cpm-app\clientsdk-webui\dist\CPMClient-2.1.war
appc:
  •
  • output the same as Step 5, Section 7
  •
all:
BUILD SUCCESSFUL
Total time: 5 minutes 4 seconds
cpm-app\clientsdk-webui >
```
8. **Deploy Sample Avaya CPM Web Client Application to BEA Weblogic AS**

These Application Notes assume that the BEA Weblogic AS is already installed. In addition, the administrative server for the domain must be operational prior to running the deploy, redeploy or undeploy targets. These Ant tasks will fail, if the administrative server is not running.

This section describes the steps used to deploy the CPM SDK client web application as a WAR file to a running BEA Weblogic AS domain using Ant. The steps described in this section include:

- Adding the deploy, redeploy, and undeploy targets to the build.xml file.
- Running the all target, to completely rebuild, the CPM client web application, after the build.xml is complete.
- Running the redeploy target after recompiling the web application.

The last two steps above are required due to modifications of the build.xml file, in sections 7 and 8.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Create a new target named <code>deploy</code> target that includes the <code>wldeploy</code> Ant task, with the <code>action</code> set equal to <code>deploy</code>. The <code>deploy</code> target tells the specified BEA Weblogic Domain to dynamically deploy the web application, and to make it available for serving user requests. The web application remains deployed across restarts of the BEA Weblogic Domain. Add the text shown in the following screenshot to the <code>build.xml</code> for the <code>deploy</code> target.</td>
</tr>
</tbody>
</table>

```
<target name="deploy" description="Deploys (and redeploys) the entire src application">
  <wldeploy user="${user}" password="${password}" adminurl="${adminurl}" action="deploy" source="${war}" verbose="${verbose}" debug="${debug}"
    name="${app.name}-${app.version}" />
</target>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Run the <strong>deploy</strong> target, and verify that the WAR file is successfully deployed. The following screenshot shows the output from running the <strong>deploy</strong> target. If the deployment fails, check the command output for error messages.</td>
</tr>
</tbody>
</table>

```bash
cpm-app\clientsdk-webui > ant deploy
Buildfile: build.xml
deploy:
  [wldeploy] weblogic.Deployer -debug -verbose -upload -noexit -name CPmClient-2.1 -source cpm-app\clientsdk-webui\dist\CPmClient-2.1. war -adminurl iiop://<BEA Weblogic AS IP Address>:7001 -user weblogic -password ******** -deploy
  ●
  ●
  ●
[wldeploy] [BasicOperation.execute():425] : Initiating deploy operation for app, CPmClient-2.1, on targets:
[wldeploy] Target state: deploy completed on Server examplesServer
[wldeploy]
[wldeploy] Target Assignments:
[wldeploy] + CPmClient-2.1 examplesServer
[wldeploy] [ServerConnectionImpl.close():334] : Closing DM connection
[wldeploy] [ServerConnectionImpl.close():354] : Unregistered all listeners
[wldeploy] [ServerConnectionImpl.closeJMX():374] : Closed JMX connection
[wldeploy] [ServerConnectionImpl.closeJMX():386] : Closed Runtime JMX connection
[wldeploy] [ServerConnectionImpl.closeJMX():398] : Closed Edit JMX connection
BUILD SUCCESSFUL
Total time: 47 seconds
cmp_app\clientsdk-webui>
```
3. Create a new target named **redeploy** that includes the **wldeploy** Ant task, with the **action** set equal to **redeploy**. The **redeploy** target tells the specified BEA Weblogic Domain to dynamically **redeploy** a new copy of the web application to the Domain. This can be useful when the web application has been modified. For example, when classes or property files have been updated in the /WEB-INF/classes directory, or when jar files have been added or modified in the /WEB-INF/lib directory.

Add the text shown in the following screenshot to the **build.xml** for the **redeploy** target.

```
<target name="redeploy" description="Redeploys just the web application">
  <wldeploy user="${user}" password="${password}" adminurl="${adminurl}" action="redeploy"
    verbose="${verbose}" debug="${debug}" name="${app.name}-${app.version}" />
</target>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>Run the <strong>redeploy</strong> target, and verify that it successfully redeploys the WAR file. The following screenshot shows the output from running the <strong>redeploy</strong> target. If the redeployment fails, check the command out for error messages.</td>
</tr>
</tbody>
</table>

```shell
cpm-app\clientsdk-webui >
cpm-app\clientsdk-webui > ant redeploy
Buildfile: build.xml
redeploy:
    ●
    ●
    ●
    [wldeploy] [BasicOperation.execute():425] : Initiating redeploy operation for app, CPMClient-2.1, on targets:
    [wldeploy] [RedeployOperation.initializeTask():55] : Starting task with path: null
    [wldeploy] Target state: redeploy completed on Server examplesServer
    [wldeploy] [ServerConnectionImpl.close():334] : Closing DM connection
    [wldeploy] [ServerConnectionImpl.close():354] : Unregistered all listeners
    [wldeploy] [ServerConnectionImpl.closeJMX():374] : Closed JMX connection
    [wldeploy] [ServerConnectionImpl.closeJMX():386] : Closed Runtime JMX connection
    [wldeploy] [ServerConnectionImpl.closeJMX():398] : Closed Edit JMX connection
BUILD SUCCESSFUL
Total time: 14 seconds
cpm-app\clientsdk-webui>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5.   | Create a new target named **undeploy** target that includes the **wldeploy** Ant task, with the **action** set equal to **undeploy**. The **undeploy** target tells the specified BEA Weblogic Domain to dynamically stop the web application from serving user requests, and to undeploy it from Domain.  
Add the text shown in the following screenshot to the **build.xml** for the **undeploy** target. |

```xml
<target name="undeploy" description="Undeploys the entire web application">
  <wldeploy user="${user}" password="${password}" adminurl="${adminurl}" action="undeploy" name="${app.name}-${app.version}" />
</target>
```

| 6.   | Run the **undeploy** target, and verify that the WAR file is successfully removed from the targeted Domain. The following screenshot shows the output from running the **undeploy** target. If the undeployment fails, check the command out for error messages. |

```bash
cpm-app\clientsdk-webui > 
cpm-app\clientsdk-webui > ant undeploy  
Buildfile: build.xml  
undeploy:  
[wldeploy] Target state: undeploy completed on Server examplesServer  
[wldeploy]  
BUILD SUCCESSFUL  
Total time: 9 seconds  
_cpm-app\clientsdk-webui > 
```

| 7.   | Run the **all** target (not shown) to recompile the CPM client web application, with the final version of the **build.xml** file. Verify that the **all** target successfully completes. |

| 8.   | Run the **redeploy** target (not shown) to deploy the recompiled web application from the previous step to the BEA Weblogic AS. |
### 9. Verification Steps

These Application Notes were confirmed using the steps below to verify that the sample Avaya CPM Client web Client was deployed correctly, and was operational on the BEA Weblogic AS.

- Verify that the sample Avaya CPM Web Client application is deployed to the BEA Weblogic AS.
- 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 & BEA Weblogic log files do not show any errors.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Verify that the CPM Client is deployed to the BEA Weblogic AS. Launch a web browser, and enter the URL <code>http://&lt;BEA Weblogic AS server IP address&gt;:7001/CPMClient-2.1</code>. Verify that the main Communications Process Manager Client web page opens.</td>
</tr>
</tbody>
</table>

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

#### CPM Services
- Advisory Service
- Notify and Conference
- Notify and Respond
- Find and Call
### Step 2
Click on the **Advisory Service** link in the **Communications Process Manager Client** window. Verify that the **Advisory Service** web page opens, and correctly displays. Fill in the fields with appropriate values. Verify that the recipients receive and acknowledge the advisory. A minimum of two users need to be created using the steps in Section 4, to perform these verifications.

#### Advisory Service

<table>
<thead>
<tr>
<th>CPM HostIP</th>
<th>CPM User Name</th>
<th>CPM Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>123.123.123</td>
<td>jenny</td>
<td>****</td>
</tr>
</tbody>
</table>

**Originator**: jenny  
**Recipients**: jenny  
**Authentication**: 

**Subject**: Weather Advisory  
**Message**: This is a severe weather advisory

- [Start Notification]  
- [Cancel Notification]  
- [Update Message]  
- [Add Recipients]  
- [Get Notification Status]

**Advisory Successfully Started.**

### Step 3
Repeat Step 2 only clicking on the **Notify and Conference** link in the **Communications Process Manager Client** web page. Verify that the Notify and Conference service can be launched (not shown). Verify that the Notify and Conference service successfully completes.

### Step 4
Repeat Step 2 only clicking on the **Notify and Respond** link in the **Communications Process Manager Client** web page. Verify that the Notify and Respond service can be launched (not shown). Verify that the Notify and Respond service is successfully completed.

### Step 5
Repeat Step 2 only clicking on the **Find and Call** link in the **Communications Process Manager Client** web page. Verify that the Find and Call service can be launched (not shown). Verify that the Find and Call service is successfully completed.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>Launch a web browser and enter the URL [http://&lt; CPM server IP address &gt;/VIA](http://&lt; CPM server IP address &gt;/VIA) to open the CPM server login page. Enter the user credentials for the CPM user account used in the above verification steps. The <strong>Communications Process Manager Inbox</strong> web page opens. Click <strong>Outbox</strong> and then select <strong>Completed</strong> from the <strong>Select</strong> drop down list 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.</td>
</tr>
</tbody>
</table>

![Communications Process Manager Screen](image_url)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</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>- nafsvc.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>
8. BEA Weblogic AS’s logging is available via the BEA Weblogic administration console web interface. To view BEA Weblogic logs select **Diagnostics → Log Files**. The **Summary of log files** opens in the right frame. Select the radio button for the log file of interest, and then click **View**.

![BEA Weblogic AS's Logging Interface](image_url)

### Log Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainLog</td>
<td>Domain Log</td>
<td>exampleServer</td>
</tr>
<tr>
<td>EventDataArchive</td>
<td>Instrumentation</td>
<td>exampleServer</td>
</tr>
<tr>
<td>JMSMessageLog/exampleServer</td>
<td>JMS Log</td>
<td>exampleServer</td>
</tr>
<tr>
<td>JMSMessageLog/ExampleServer</td>
<td>JMS Log</td>
<td>exampleServer</td>
</tr>
<tr>
<td>JMSMessageLog/MessageServer</td>
<td>JMS Log</td>
<td>exampleServer</td>
</tr>
<tr>
<td>ServerLog</td>
<td>Server Log</td>
<td>exampleServer</td>
</tr>
</tbody>
</table>
Step 9.
The selected log file opens in the right frame for viewing. All entries in the log file can be searched, by using the Previous and Next selections at the top of the log entries table. Select the radio button for the event of interest, and click View, to view the event details (not shown).
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.</td>
<td>The <code>servername.log</code>, <code>domainname.log</code>, and <code>access.log</code> files also are available in the <code>…/domains/domainName/servers/serverName/logs</code> directory on the server. The BEA Weblogic administration console is not required to view these logs. The log files located in this directory can be opened for viewing using any text editor on the server.</td>
</tr>
<tr>
<td>11.</td>
<td>Debug messages can be written to the log files. In the <strong>Domain Structure</strong> frame select the <strong>Environment</strong> → <strong>Servers</strong> item to open the <strong>Summary of Servers</strong> frame on the right (not shown). In the <strong>Name</strong> column of the servers table click on the <strong>server link</strong> of interest (e.g., <code>examplesServer</code>).</td>
</tr>
<tr>
<td>12.</td>
<td>The <strong>Settings for server</strong> (e.g., <code>examplesServer</code>) frame opens on the right. At the top of the frame select the <strong>Debug</strong> tab to display the <strong>Debug settings for the Server</strong> table. In the <strong>Change Center</strong> frame click the <strong>Lock &amp; Edit</strong> button, to activate the fields on this page. Then check the <strong>Debug Scopes and Attributes</strong> of interest. Checking the checkbox in the title row of the table selects all the debug scopes. Click the <strong>Enable</strong> button to enable logging of debug messages.</td>
</tr>
<tr>
<td>13.</td>
<td>Click the <strong>Activate Changes</strong> button in the <strong>Change Center</strong> frame (not shown) to activate the logging of debug messages.</td>
</tr>
<tr>
<td>14.</td>
<td>Repeat the above steps only clicking <strong>Disable</strong> to disable the logging of debug messages.</td>
</tr>
</tbody>
</table>
10. Conclusion

These Application Notes show how to build and deploy the sample Avaya CPM Client web application on a BEA Weblogic AS using BEA Weblogic development tools and Apache Ant. In addition, it is shown that the Web Applications deployed on a BEA Weblogic AS interoperate with Avaya Communications Process Manager via web services.

11. Additional References

The following Avaya product documentation is available at http://support.avaya.com.


The following documents may be obtained from http://e-docs.bea.com/wls/docs100/ and http://e-docs.bea.com/wlw/docs100/.


12. Appendix A: Build.xml Listing

The following is a listing of the contents of the build.xml file used in these Application Notes. This build.xml file incorporates all the changes made in these Application Notes.

```xml
<?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 application.
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=".">

<!-- ===================== Property Definitions =========================== -->

<!-- Each of the following properties are used in the build script. Values for these properties are set by the first place they are defined, from the following list:

* Definitions on the "ant" command line (ant -Dfoo=bar compile).
* Definitions from a "build.properties" file in the top level source directory of this application.
* Definitions from a "build.properties" file in the developer's home directory.
* Default definitions in this build.xml file.

You will note below that property values can be composed based on the
```
contents of previously defined properties. This is a powerful technique that helps you minimize the number of changes required when your development environment is modified. Note that property composition is allowed within "build.properties" files as well as in the "build.xml" script.

-->  

<property file="build.properties"/>
<property file="${user.home}/build.properties"/>
<property file="../../build.properties"/>

<!-- ----------- 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".

manager.password The login password of a user that is assigned the "manager" role (so that he or she can execute commands via the "/manager" web application)

manager.url   The URL of the "/manager" web application on the Tomcat installation to which we will deploy web applications and web services.

manager.username The login username of a user that is assigned the "manager" role (so that he or she can execute commands via the "/manager" web application)

-->  

<property name="app.name" value="CPMClient"/>
<property name="app.path" value="/\{app.name\}"/>
<property name="build.home" value="\{basedir\}/build"/>
<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"/>

<!-- The following lines added based upon the weblogic build.xml template. -->

<property name="war" value="${dist.home}/${app.name}-${app.version}.war"/>
<property name="debug" value="true"/>
<property name="verbose" value="true"/>
<property name="user" value="weblogic"/>
<property name="password" value="weblogic"/>
<property name="servername" value="examplesServer"/>
<property name="adminurl" value="iiop://135.8.139.229:7001"/>

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

<!-- Dummy external dependency -->
<!--
<property name="foo.jar" value="/path/to/foo.jar"/>
-->

<!-- ============== Compilation Classpath =============== -->

<!-- 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 *** -->
<!--
<fileset dir="${src.lib}"
<include name="**/*.jar"/>
</fileset>
<!-- Prepare Target ----------------------------------------- -->

<!-- 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. -->

<target name="prepare">

<!-- Create build directories as needed -->
<mkdir dir="${build.home}"/>
<mkdir dir="${build.home}/WEB-INF"/>
<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 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="${build.home}/WEB-INF/classes"/>
    <wlcompile srcdir="${src.home}" destdir="${build.home}/WEB-INF/classes">
        <classpath refid="compile.classpath"/>
    </wlcompile>
    <!-- Copy application resources -->
    <copy todir="${build.home}/WEB-INF/classes/com">
        <fileset dir="${build.home}/WEB-INF/classes/APP-INF/classes/com"/>
    </copy>
    <delete dir="${build.home}/WEB-INF/classes/APP-INF"/>
    <copy todir="${build.home}/WEB-INF/classes/com/avaya/cpm/clientsdk/wui" file="${src.home}/com/avaya/cpm/clientsdk/wui/ApplicationResources.properties"/>
</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,appc" 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="Package a standard J2EE WAR for distribution">
  <!-- Copy documentation subdirectories -->
  <mkdir dir="${dist.home}/docs"/>
  <copy todir="${dist.home}/docs">
    <fileset dir="${docs.home}"/>
  </copy->

  <wlpackage srcdir="${src.home}" destdir="${build.home}" toFile="${war}"/>

  <!-- Copy additional files to ${dist.home} as necessary -->
</target>

<!-- =============== appc Target =============== -->
<!--
    The "appc" target generates JSPs and container-specific EJB classes for deployment, and validates deployment descriptors for compliance with the current Java EE specifications. appc performs validation checks between the application-level deployment descriptors and the individual modules in the application as well as validation checks across the modules.
-->
<target name="appc" depends="dist" description="Runs weblogic.appc on your application">
  <wlappc source="${war}" forceGeneration="true" debug="${debug}" verbose="${verbose}"/>
  <classpath refid="compile.classpath"/>
</wlappc>
</target>

<!-- =============== Source Dist 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

<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 name="**/*"/>
      <exclude name="ant-build/**"/>
      <exclude name=".settings/**"/>
      <exclude name="build/**"/>
      <exclude name="dist/**"/>
      <exclude name="*.svn/**"/>
    </fileset>
  </zip>
  <!-- Copy additional files to ${dist.home} as necessary -->
</target>

<!-- ==================== Javadoc Target ================================== -->

<!-- The "javadoc" target creates Javadoc API documentation for the Java classes included in your application. Normally, this is only required when preparing a distribution release, but is available as a separate target in case the developer wants to create Javadocs independently. -->

<target name="javadoc" depends="compile" description="Create Javadoc API documentation">
  <mkdir dir="${dist.home}/docs/api"/>
  <javadoc sourcepath="${src.home}"
    destdir="${dist.home}/docs/api"
    packagenames="*">
    <classpath refid="compile.classpath"/>
  </javadoc>
</target>

<!-- ==================== Deploy Target ================================== -->

<!-- The "Deploy" target tells the specified BEA Weblogic Domain to dynamically deploy this web application, and to make it available for serving user requests. The web application remains deployed accross restarts of the BEA Weblogic Domain. -->

<target name="deploy" description="Deploys (and redeploys) the entire src application">
  <wldeploy user="${user}" password="${password}" adminurl="${adminurl}"
    action="deploy" source="${war}" upload="true"
    verbose="${verbose}" debug="${debug}"/>
<!-- Redeploy Target -->

The "Redeploy" target tells the specified BEA Weblogic Domain to dynamically undeploy the web application from the Domain, and deploys a new copy of the web application to the Domain.

This can be useful when the web application has been modified. For example, when classes or property files have been updated in the /WEB-INF/classes directory, or when jar files have been added or modified in the /WEB-INF/lib directory.

<!-- Undeploy Target -->

The "Undeploy" target tells the specified BEA Weblogic Domain to dynamically stop the web application from serving requests, and to undeploy it from Domain.

</project>
©2008 Avaya Inc. All Rights Reserved.

Avaya and the Avaya Logo are trademarks of Avaya Inc. All trademarks identified by ® and ™ are registered trademarks or trademarks, respectively, of Avaya Inc. All other trademarks are the property of their respective owners. The information provided in these Application Notes is subject to change without notice. The configurations, technical data, and recommendations provided in these Application Notes are believed to be accurate and dependable, but are presented without express or implied warranty. Users are responsible for their application of any products specified in these Application Notes.

Please e-mail any questions or comments pertaining to these Application Notes along with the full title name and filename, located in the lower right corner, directly to the Avaya Solution & Interoperability Test Lab at interoplabnotes@list.avaya.com