Avaya Solution & Interoperability Test Lab

Application Notes for Packaging and Deploying Avaya Communications Process Manager Sample SDK Web Application on a JBoss Application Server – Issue 1.0

Abstract

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager sample Software Development Kit web application on a JBoss Application Server.

JBoss is an application server program for use with Java 2 Platform, Enterprise Edition (J2EE) and Enterprise Java Beans (EJB). For these Application Notes, a sample Avaya Communications Process Manager web client application available in the CPM SDK was deployed to the JBoss Application Server. This client application was used to verify the Avaya CPM Simple Object Access Protocol (SOAP) web services interaction between the Avaya CPM and JBoss Application Server platform.

The sample configuration used to validate the integration consists of Avaya S8710 Servers with a G650 Media Gateway running Avaya Communication Manager, Avaya SIP Enablement Services, Avaya Voice Portal, Nuance Text to Speech server, Avaya Meeting Exchange Express, Avaya Communications Process Manager, and the JBoss Application Server.

©2008 Avaya Inc. All Rights Reserved.

1. Introduction

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager sample SDK web application on a JBoss Application Server

JBoss is an application server program 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 deployed to the JBoss Application Server. This web client application was used to verify the CPM SOAP Web services interaction between the Avaya CPM and JBoss Application Server platform. 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 the JBoss Application Server.

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 [8] in the Additional References section. For these Application Notes, the Avaya CPM SDK web client application was used to verify the interoperability between the JBoss Application Server and the Avaya CPM via web services.

©2008 Avaya Inc. All Rights Reserved.

2. Configuration

Figure 1 provides an overview of the network used in the sample configuration. This sample configuration consists of Avaya S8710 Servers with a 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 the JBoss Application Server. The Avaya 9600 Series H.323 and SIP Telephones are registered with Avaya Communication Manager and Avaya SIP Enablement Services (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.

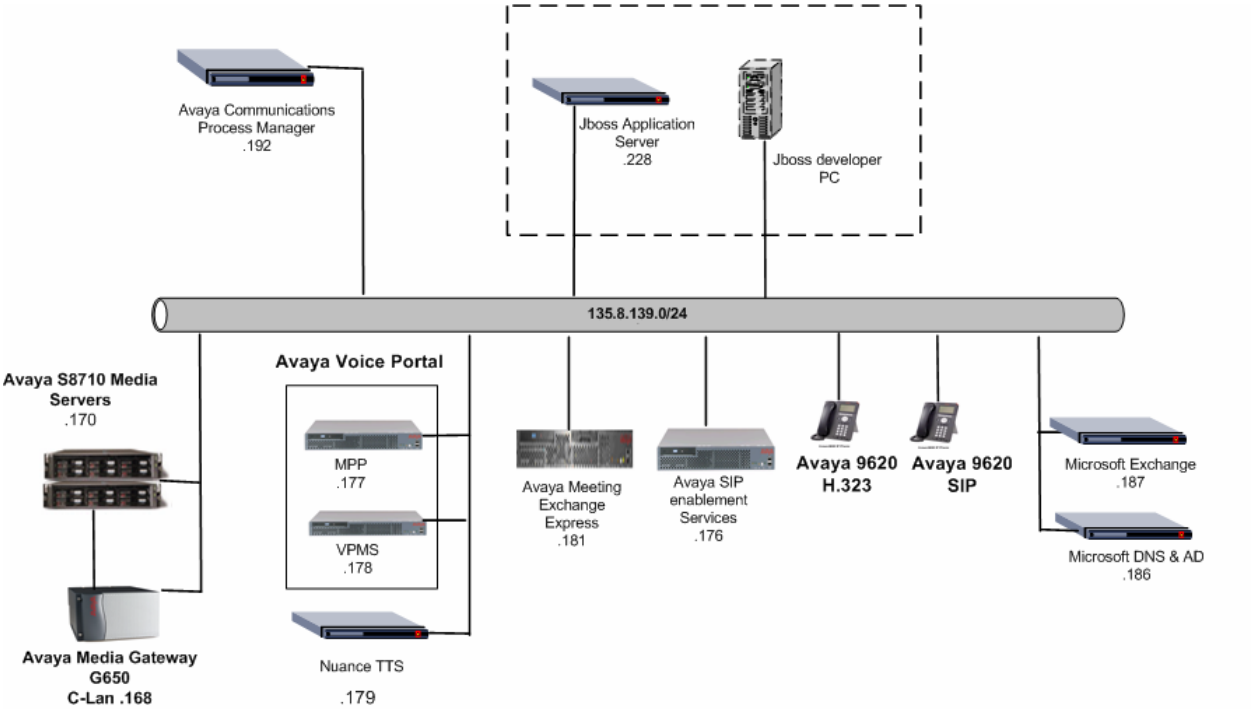


Figure 1: Network Configuration

©2008 Avaya Inc. All Rights Reserved.

3. Equipment and Software Validated

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

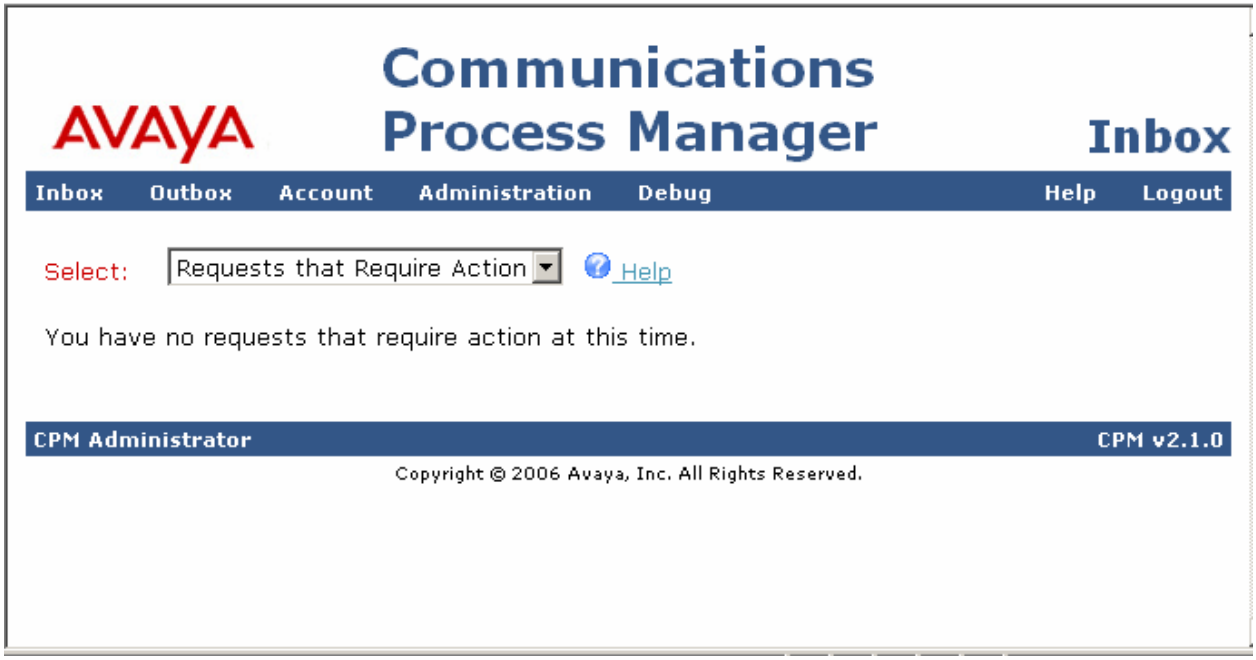
Equipment & Software	Version
Avaya S8710 Servers	Avaya Communication Manager 4.0.1 (R014x.00.1.731.2)
Avaya G650 Media Gateway TN2302AP MEDPRO TN799DP CLAN	 HW20 FW116 HW01 FW024
Avaya Meeting Exchange Express	2.5.60.0
Avaya SES Enablement Services	4.0-04.0.033.6
Avaya Voice Portal • VPMS • MPP	4.0.0.0.2901
Nuance TTS server Real speak	 4.0.10
Avaya 9600 Series SIP Telephones	1.0.2.2
Avaya 9600 Series H323 Phones	1.5
Avaya Communications Process Manager	Release 2.1
Avaya CPM SDK	Release 2.1.54
JBoss Application Server • Microsoft Windows 2003 Server • Sun Microsystems Java JDK	Release 4.2.2 Service Pack 2 1.5_0_13
Windows XP Professional • Sun Microsystems Java JDK	2002 SP 2 1_5_13
Microsoft Active Directory and DNS Server Microsoft Windows Server 2003	Service Pack 2
Apache Software Foundation Axis	1.4
Apache Software Foundation Ant	1.7.0

Table 1: Equipment/Software List

©2008 Avaya Inc. All Rights Reserved.

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 [1], [2], [3], [4], and [5] in the Additional References section. 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.

Step	Description
1.	On the Avaya CPM server, launch a web browser and enter the URL **http://<Name or IP address of CPM server>/VIA**. When prompted for a user name and password, enter the credentials of the administrator account.
2.	The Communication Process Manager Administration Portal page appears. To add a new user, click **Account**. 

©2008 Avaya Inc. All Rights Reserved.

Step	Description
3.	The **Account Home** page appears. To create a new user account, for CPM Notifications and Response system, click **Create Account**.

©2008 Avaya Inc. All Rights Reserved.

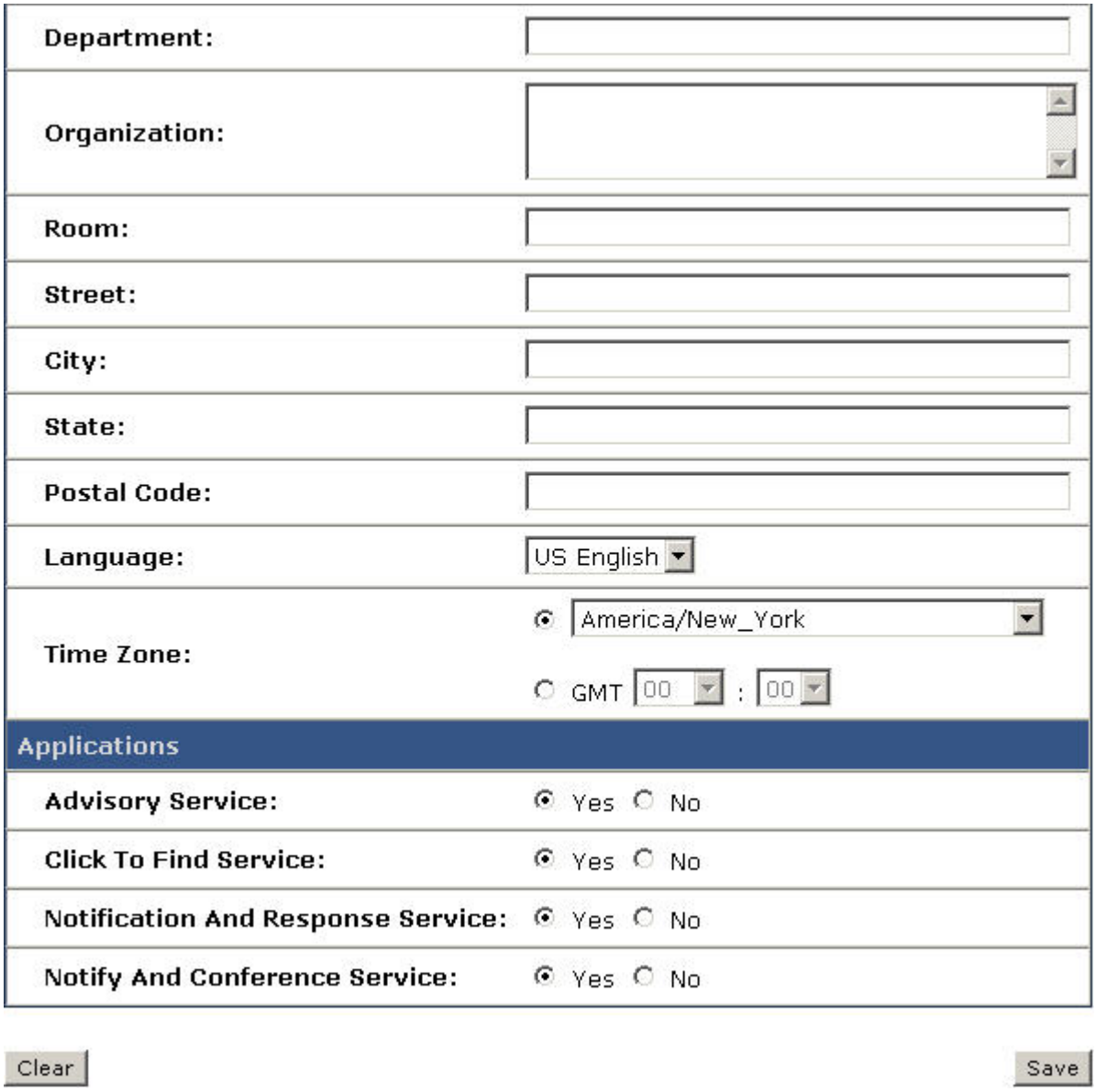
Step	Description
4.	The **Create Account** screen appears. Enter the following values: • **Administrator**: To give the user administrator access, select **Yes**. • **CPM User**: To make this user a licensed Communications Process Manager user, select **Yes**. Licensed Communications Process Manager Users can log in to the Communications Process Manager portal and receive notifications. • **Handle**: Enter the user's handle (e.g., **Bill Smith**). • **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**. The **Create Account** screen is shown in **Steps 5** and **6**.

©2008 Avaya Inc. All Rights Reserved.

Step	Description
5.	

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
Manager:	

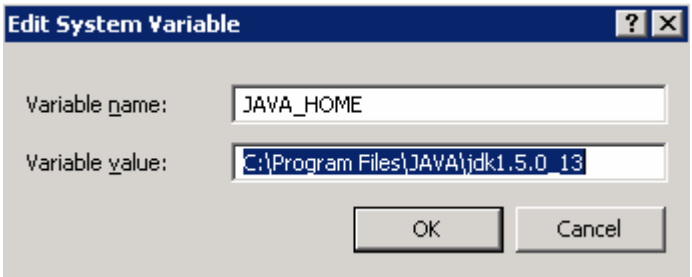
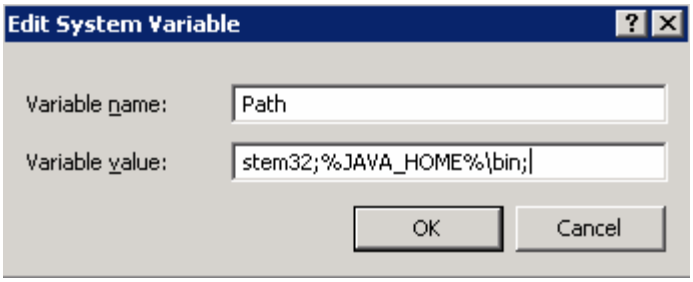
©2008 Avaya Inc. All Rights Reserved.

Step	Description
6.	
7.	Repeat **Step 4** to create additional CPM users as necessary.

©2008 Avaya Inc. All Rights Reserved.

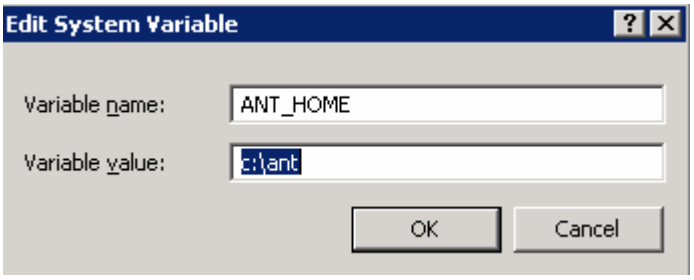
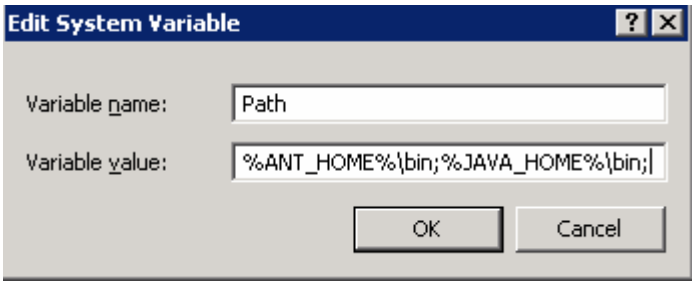
5. Compile Sample Avaya CPM Web Client Application

This section illustrates how to compile the Avaya CPM SDK web client application running on a Windows XP PC with Service Pack 2. These Application Notes assume that a proper version of Java Developers Kit (JDK) is already installed on the PC. The CPM Client SDK can be downloaded from the following URL: http://devconnect.avaya.com/. For more information on the sample Avaya SDK web client application refer to [7] in the Additional References section. Apache Ant is a Java build tool. In this sample configuration, Ant is used to compile the client application source code and build the CPM client application war file[1]. Ant can be downloaded from the following URL: http://ant.apache.org.

Step	Description
1.	Verify that the JAVA_HOME environment variable points to the JDK installation directory. Click **Start → Control Panel → System → Advanced → Environment Variables**. In the System variables pane, verify that the JAVA_HOME is properly configured. In these Application Notes, the Java JDK5.0_13 is installed in c:\Program files\JAVA\jdk1.5.0_13. The result is shown below.  In the System variables pane, click **Path → Edit** to verify that **"%JAVA_HOME%\bin**;" is defined in the **Variable_value** field for the **Path Variable name** 
2.	Download and unzip Apache Ant (e.g., **C:\ant**).

[1] The "war" file shipped with the Avaya CPM SDK can be deployed out of the Box on the JBoss Application Server.

©2008 Avaya Inc. All Rights Reserved.

Step	Description
3.	Add the ANT_HOME environment variable. Click **Start → Control Panel → System → Advanced → Environment Variables**. In the System variables pane, add "ANT_HOME" and set the **Variable_value** value to the Ant installation directory. In these Application Notes, ANT is installed under "c:\ant". The result is shown below.  In the **System variables** pane, click **Path → Edit**. Add **"%ANT_HOME%\bin;**" in the **Variable_value** field. 
4.	The Avaya CPM SDK web client application comes with the source code. Unzip the CPM SDK (e.g., **C:\CPMSDK2.1**). In this sample configuration, the SDK install directory is referred as <CPM-SDK-DIR>.

5.	Clean the CPM client build environment.
	• Start a Windows Command Prompt.
	• Change directory to C:\<CPM-SDK-DIR>\javasdk\clientsdk-webui.
	• From the command prompt, run "ant clean".
	• Verify that the "BUILD SUCCESSFUL" message is displayed on the screen.
	`C:\>cd C:\CPMSDK2.1\javasdk\clientsdk-webui`
	
	`C:\CPMSDK2.1\javasdk\clientsdk-webui>`
	`C:\CPMSDK2.1\javasdk\clientsdk-webui>ant clean`
	`Buildfile: build.xml`
	
	`clean:`
	
	`BUILD SUCCESSFUL`

©2008 Avaya Inc. All Rights Reserved.

Step	Description
6.	The build.xml file contains information that the Java Ant task uses to build the sample CPM SDK web application into CPMClient-2.1.50.war archive file. Refer to Appendix A for the build.xml file.

7.	Build the CPM client build environment.
	• From the command prompt, run "ant build".
	• Verify that the "BUILD SUCCESSFUL" message is displayed on the screen.
	`C:\CPMSDK2.1\javasdk\clientsdk-webui>`**`ant compile`**
	`Buildfile: build.xml`
	`…`
	`compile:`
	`  [javac] Compiling 9 source files to C:\CPMSDK2.1\javasdk\clientsdk-webui\bui`
	`ld\WEB-INF\classes`
	`  [copy] Copying 2 files to C:\CPMSDK2.1\javasdk\clientsdk-webui\build\WEB-IN`
	`F\classes`
	`BUILD SUCCESSFUL`

8.	Run "ant dist" to create the CPM client web application archive (war) file. • From the command prompt, run "ant dist". • Verify that the "BUILD SUCCESSFUL" message is displayed on the screen. • Verify that the "CPMClient-2.1.war" file is created in the **dist** directory. `C:\CPMSDK2.1\javasdk\clientsdk-webui>`**`ant dist`** `Buildfile: build.xml` **`BUILD SUCCESSFUL`**

©2008 Avaya Inc. All Rights Reserved.

6. Deploy a Sample Avaya CPM Client Application to JBoss Application Server

JBoss Application Server is the open source implementation of the Java EE suite of services. These Application Notes assume that the JBoss Application Server is already installed on a Microsoft Windows 2003 server. Refer to [9] in the Additional References section for the JBoss Application Server installation.

Note: For the Avaya CPM-JBoss integration, the installation of the JBoss Java Message Service (JMS) is not required. These Application Notes further assume that a proper version of the Sun JDK is downloaded and configured.

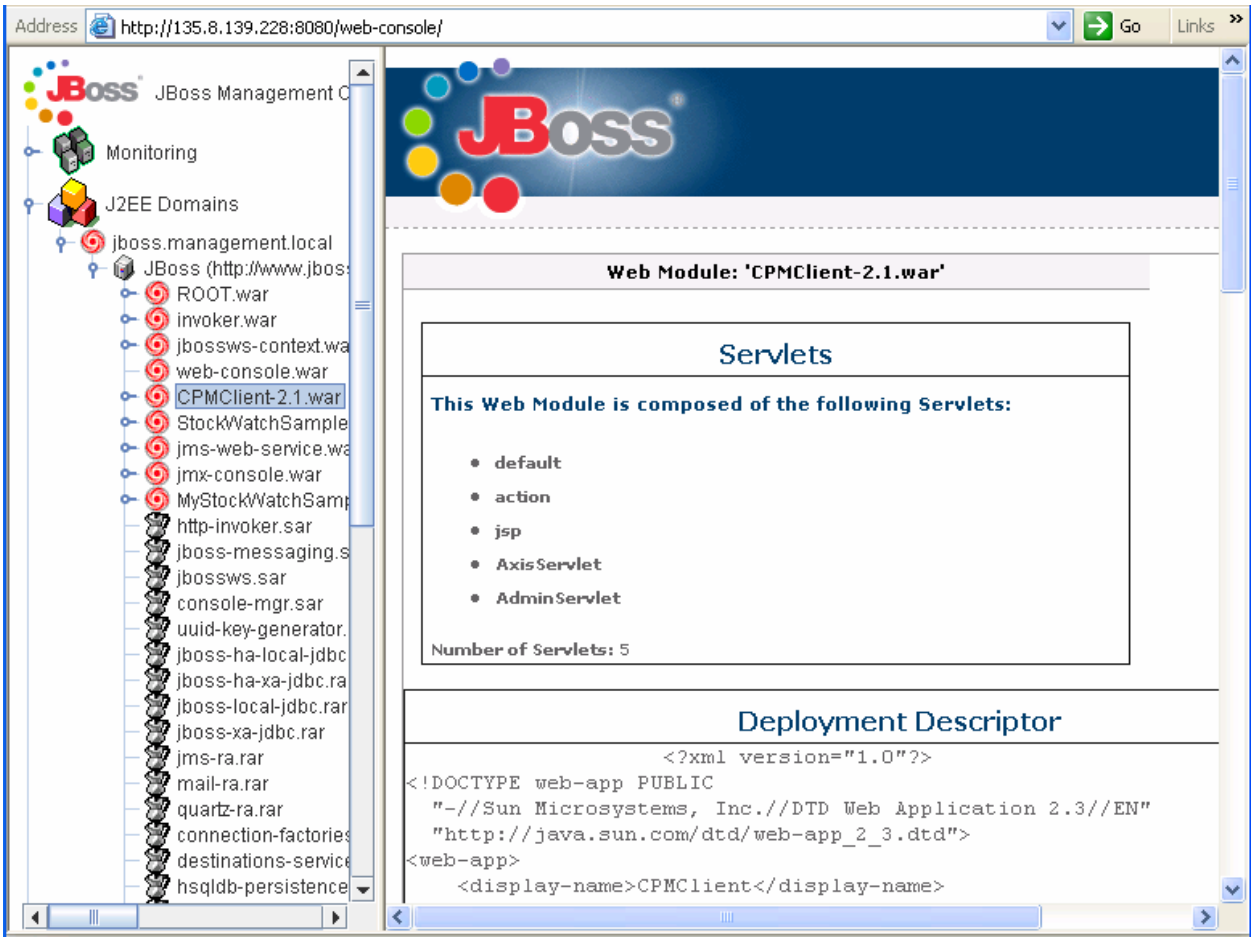
This section illustrates how to deploy the CPM SDK client application to the JBoss Application Server. In this sample configuration, JBoss messaging was installed and was configured to run with JMS. When JBoss is running JMS, the CPM client application has to be deployed to the following directory: <JBoss-Installed-Dir>\server\messaging\deploy

Step	Description
1.	Copy the CPMClient-2.1.war to the JBoss deploy directory: **C:\ <JBoss-Installed-Dir>\server\messaging\deploy**. The JBoss AS will automatically deploy the application. **Note**: It may take up to 4 minutes for the application to start up.

©2008 Avaya Inc. All Rights Reserved.

Step	Description
2.	Launch a web browser and enter the URL **http://<Name or IP address of JBoss server>:8080/** The **JBoss** screen appears.

©2008 Avaya Inc. All Rights Reserved.

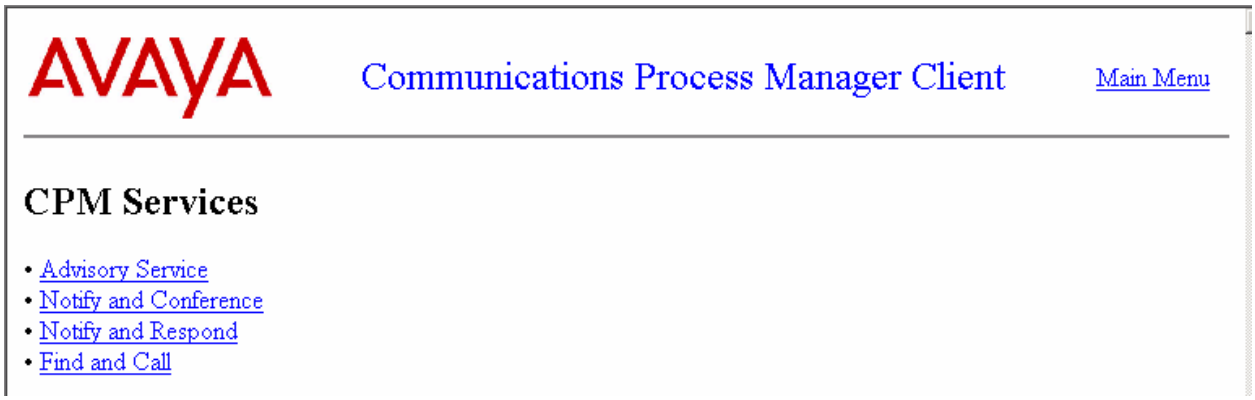
Step	Description
3.	Verify that the CPMclient-2.1.war is deployed to the Sun-AS. Click on **JBoss Web Console → J2EE Domains → jboss.management.local → CPMClient-2.1.war**. The **Web Module: 'CPMClient-2.1.war'** screen is shown below. 

©2008 Avaya Inc. All Rights Reserved.

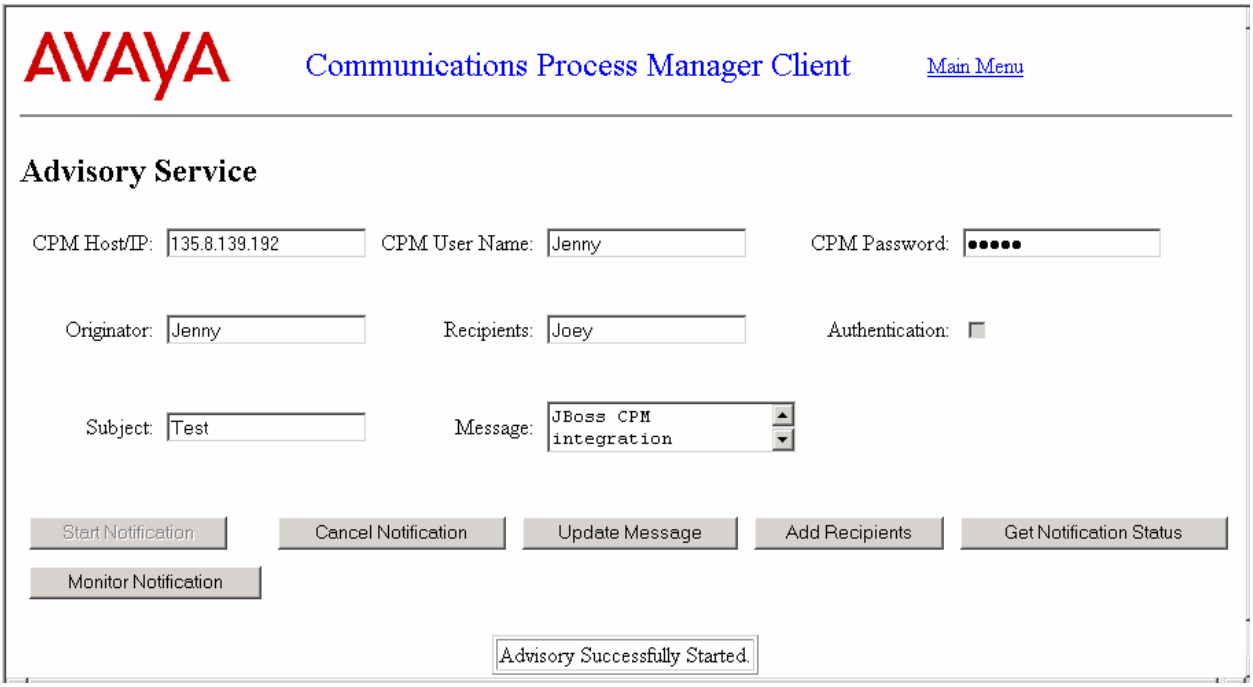
7. Verification Steps

The following steps may be used to verify proper configuration of Avaya CPM and JBoss AS:

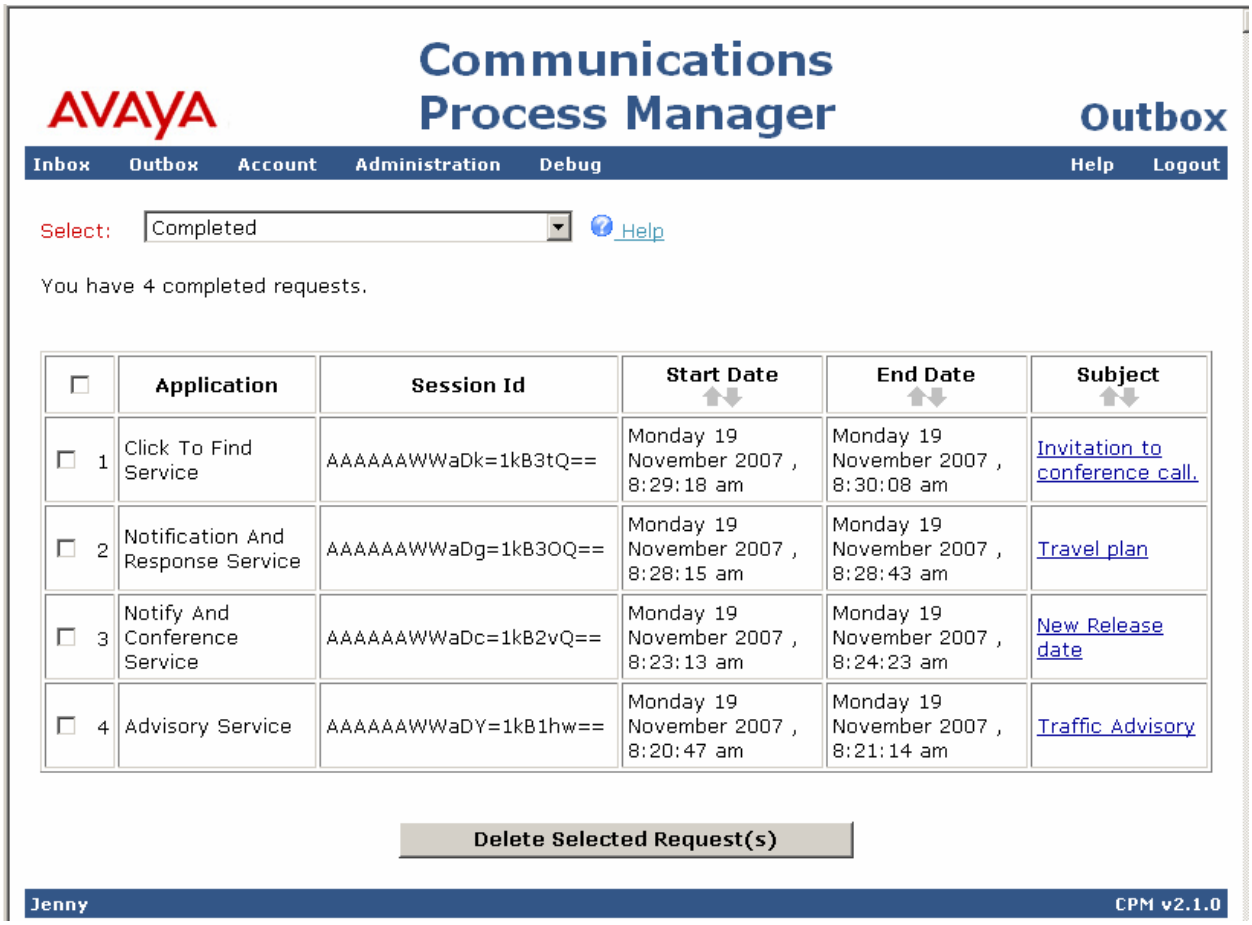
- Verify that the sample Avaya CPM Web Client application is deployed to the JBoss Application Server.
- Verify that the CPM Advisory service can be launched and acknowledged by all the recipients.
- Verify that CPM Notify and Conference service can be launched and the recipients can join the conference.
- 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 & JBoss log files do not show any errors.

Step	Description
1.	Verify that the CPMClient web application can be started from the JBoss Application Server. • Launch a web browser, enter the URL **http://<IP address of JBoss AS server>:8080/CPMClient-2.1**. • Verify that the **Communications Process Manager Client** screen appears. 

©2008 Avaya Inc. All Rights Reserved.

Step	Description
2.	Verify that the CPM Advisory service can be launched. Verify that the recipients acknowledge receipt of the notification. 
3.	Repeat Step 2 and verify that CPM Notify and Conference service can be launched. Verify that the Notify and Conference service is successfully completed.
4.	Repeat Step 2 and verify that CPM Notify and Respond service can be launched. Verify that the Notify and Respond service is successfully completed.
5.	Repeat Step 2 and verify that CPM Find and Call service can be launched. Verify that the Find and Call service is successfully completed.

©2008 Avaya Inc. All Rights Reserved.

Step	Description
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 → **Debug** and then select the following fields to view the completed notifications. • Select "Completed" from the **Select** field to view the completed notifications. • Select **VoiceXML**. • Check **Responses**. The completed notifications are displayed as shown below. To view the contents of a notification, click the subject of the notification. 

©2008 Avaya Inc. All Rights Reserved.

8. Troubleshooting Steps

This section contains simple troubleshooting steps for debugging purposes.

Step	Description
1.	The following log files are available on the Avaya CPM for troubleshooting purposes. The log files are stored under the **/var/log/cpm** directory. • **mail.log** • **nafsvc.log** • **oam.log** • **dcore.log** • **commflow.log** • **advsvc.log** • **b2bua.log** • **account.log** • **via.log** • **platform-licensing.log** • **user-licensing.log** • **cs-tomcat-memorymonitor.log** • **cpm.log** • **complete.log**
2.	JBoss Application Server logging is configured from **`jboss-log4j.xml`**. This XML file specifies the log file name(s), categories of messages, the message format and the level of filtering. By default, the log file name is **server.log**. In the sample configuration, the **`jboss-log4j.xml`** file is under "`C:\<JBoss-Install-dir>\server\messaging\conf`" and the log file **server.log** is under "`C:\<JBoss-Install-dir>\server\messaging\log`".

©2008 Avaya Inc. All Rights Reserved.

9. Conclusion

As illustrated in these Application Notes, a JBoss Application Server 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.

[1] "Administrator Guide for Avaya Communication Manager", Issue 3.1, Doc ID: 03-300509, February 2007.

[2] "Avaya Communications Process Manager Installation and Configuration Guide Release 2.1", Issue 1, Oct 2007, Document Number 04-601158

[3] "Avaya Communications Process Manager Online help for the Administrative Web Portal Release 2.1", Issue 4, Oct 2007, Document Number 04-601163

[4] "SIP Enablement Services (SES) Implementation Guide", Issue 4, Doc ID : 16-300140, May 2007

[5] "Avaya Voice Portal 4.0 Documentation Library", Jun-2007

[6] "Avaya Meeting Exchange Express Edition Release 1.5 Installation and Configuration Guide", Issue 1, Doc ID: 04-601898, March 2007.

[7] Avaya DevConnect web site https://devconnect.avaya.com/

[8] "Avaya Communications Process Manager Release 2.1 Application Programmer's Guide", Issue 1, Doc ID: 04-602358, Oct 2007.

The following documents may be obtained from http://www.JBoss.com.

[9] "JBoss Enterprise Application Platform 4.2.2 Installation Guide", Oct 2007.

[10} "JBoss Enterprise Application Platform 4.2.2 configuration Guide", Oct 2007.

ANT 1.7.0 may be downloaded from: http://ant.apache.org.
Java1.5 may be downloaded from: http://java.sun.com/javase/downloads/index_jdk5.jsp
Axis 1.4 may be downloaded from: http://tomcat.apache.org
Java Beans Activation Framework may be downloaded from:
http://java.sun.com/products/javabeans/jaf/index.jsp/

©2008 Avaya Inc. All Rights Reserved.

11. Appendix A – build.xml file

This section shows the build.xml file as a reference on how to build the sample SDK web application into CPMClient-2.1.50.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
```


©2008 Avaya Inc. All Rights Reserved.

```
                        "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.

-->

<!-- Dummy external dependency -->
```


©2008 Avaya Inc. All Rights Reserved.

```
<!--
  <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 *** -->
<!--
    <pathelement location="${foo.jar}"/>
-->

    <!-- Include all elements that Tomcat exposes to applications -->
    <!--pathelement location="${catalina.home}/common/classes"/>
    <fileset dir="${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>

  </path>

<!-- ==================== 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"/>
```


©2008 Avaya Inc. All Rights Reserved.

```
          <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="${build.home}/WEB-INF/lib"/>
      <!--
          <copy todir="${build.home}/WEB-INF/lib" file="${foo.jar}"/>
      -->
          <copy todir="${build.home}/WEB-INF/lib">
            <fileset dir="${src.lib}">
             <include name="*.jar"/>
             <exclude name="junit-3.8.1.jar"/>
             <exclude name="catalina-ant.jar"/>
             <exclude name="servlet-api.jar"/>
             <exclude name="generated"/>
               </fileset>
                   <fileset dir="${src.lib}/generated">
                    <include name="*.jar"/>
                      </fileset>
          </copy>

          <!-- 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".

-->
```


©2008 Avaya Inc. All Rights Reserved.

```
        <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>
```


©2008 Avaya Inc. All Rights Reserved.

```
<!-- ==================== 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}-${app.version}-src.zip"
update="ture">
                <fileset dir="${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>


</project>
```


©2008 Avaya Inc. All Rights Reserved.

© 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 and filename, located in the lower right corner, directly to the Avaya Solution & Interoperability Test Lab at interoplabnotes@list.avaya.com